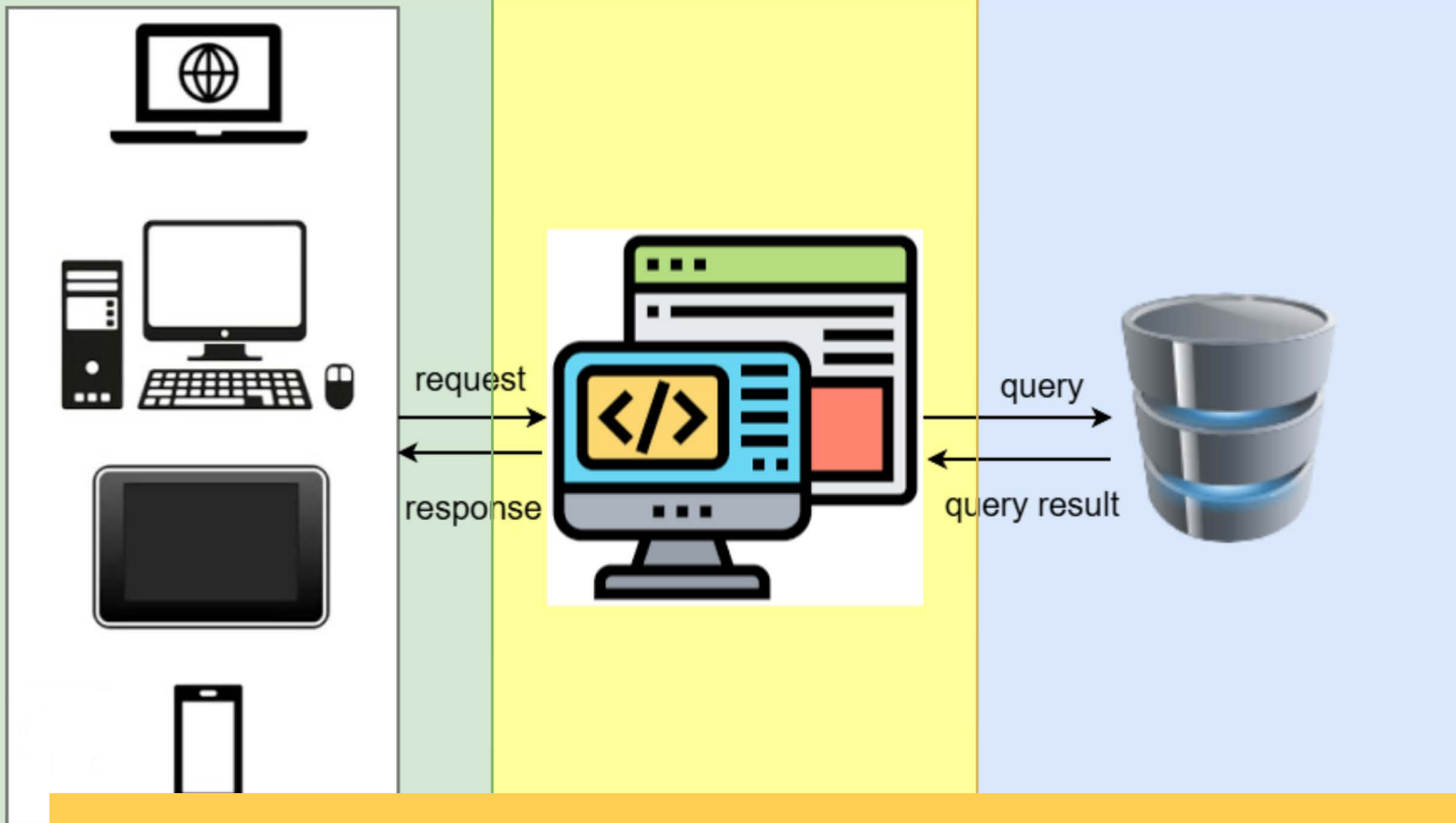




RAPPORTI TECNICI INGV

INGV Spectral Library - an enhanced spectral library for spectral pre-processing and future analytical developments



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

React JS



express

525

MySQL

**Direttore Responsabile**

Daniela VERSACE

Editor in Chief

Milena MORETTI

Editorial Board

Laura ALFONSI

Sofia BARANELLO

Carla BARNABA

Christian BIGNAMI

Simona CARANNANTE

Maria Grazia CIACCIO

Luca COCCHI

Luigi CUCCI

Lorenzo CUGLIARI

Elisabetta D'ANASTASIO

Alessia DI CAPRIO

Roberto DI MARTINO

Domenico DI MAURO

Domenico DORONZO

Filippo GRECO

Alessandro IAROCCI

Mario MATTIA

Daniele MELINI

Anna NARDI

Lucia NARDONE

Marco OLIVIERI

Arianna PESCI

Pierangelo ROMANO

Paola SBARRA

Davide SCAFIDI

Maurizio SOLDANI

Sara STOPPONI

Umberto TAMMARO

David ZULIANI

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

© INGV Istituto Nazionale di Geofisica e Vulcanologia

Rappresentante legale: Presidente INGV

Sede: Via di Vigna Murata, 605 | Roma

Tutti i diritti riservati

www.ingv.it



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

RAPPORTI TECNICI INGV

INGV Spectral Library - an enhanced spectral library for spectral pre-processing and future analytical developments

Marco Solinas*, Massimo Musacchio, Malvina Silvestri, Maria Fabrizia Buongiorno, Sergio Falcone, Angelo La Regina

INGV | Istituto Nazionale di Geofisica e Vulcanologia, Osservatorio Nazionale Terremoti

*Corresponding author

Accepted 23 April 2026 | Accettato 23 aprile 2026

How to cite | Come citare Solinas, M., Musacchio, M., Silvestri, M., Buongiorno, M.F., Falcone, S., La Regina, A. (2026). INGV Spectral Library - an enhanced spectral library for spectral pre-processing and future analytical developments. *Rapporti Tecnici INGV*, 525, 1-74 pp. <https://doi.org/10.13127/rpt/525>

Cover High-level architecture diagram of the INGV Spectral Library | In copertina Libreria spettrale INGV - una libreria spettrale avanzata per la pre-elaborazione spettrale e futuri sviluppi analitici

525

INDEX

Abstract	9
Introduction	9
1. System Overview	11
1.1 Overview and architectural principles	11
1.2 High-level diagram (Frontend $\rightleftharpoons$ Backend $\rightleftharpoons$ Database)	12
2. Architecture and Data (end-to-end view: FE $\rightarrow$ BE $\rightarrow$ DB)	12
2.1 Frontend (React)	12
2.1.1 Global Flow (overview)	13
2.1.2 Home (Card-based domains)	13
2.1.3 Explore spectral data UI (navigation hub)	14
2.1.3.1 Interactive Map	14
2.1.3.2 Tabular data exploration (Data Table Exploration)	15
2.1.4 Graph Dashboard UI	15
2.1.4.1 Reflectance (+ Add dataset)	16
2.1.4.2 Continuum removal	16
2.1.4.3 Convolution / SRF & Resampling	17
2.1.4.4 First derivative	18
2.1.5 Cross-cutting components (selectors & panels)	18
2.1.6 API client & hooks	19
2.1.7 Export, styling, and accessibility	20
2.2 Backend (Node/Express)	20
2.2.1 API surface and routing	20
2.2.2 Request-response flow and data access	21
2.2.3 Reliability, security, and observability	22
2.2.4 Configuration and deployment	22
2.3 Database (PostgreSQL)	23
2.4 GIS integration and map (OpenStreetMap)	24
2.5 Security and traceability (brief notes)	24
3. Spectral analysis tools (GUI)	25
3.1 Real-time visualisation and simultaneous comparison of spectral signatures	25
3.2 Dynamic cascading filters and reactive table	26
3.3 First Derivative	26
3.4 Continuum Removal	27
3.5 Resampling & SRF	28
4. Ingest & Data Pipeline	30
4.1 Ingest workflow: spectra and metadata	30
4.2 Supported formats and naming conventions	30

5. Validation, performance, and reliability	31
5.1 Testing methodology	31
5.2 Query performance and average endpoint response times; DB/BE scalability notes	33
5.3 Current limitations	34
6. Deployment and runtime environment	35
6.1 Runtime requirements, environment variables, and configuration	35
6.2 Repository structure, startup scripts, logging & monitoring	36
6.3 Containerisation and CI roadmap	36
6.4 Main dependencies of the application stack	37
7. Conclusions and future developments	38
Data and Software Availability	39
References	39
Sitography	40
Appendix A - API surface (FE$\rightleftharpoons$BE contract)	43
A.1 Main endpoint table	45
A.2 Call examples	46
A.2.1 Sample spectrum (Graph Dashboard)	46
A.2.2 Additional spectrum for comparison (Add dataset)	46
A.2.3 Map markers (geographic overview)	46
A.2.4 Rows for the Data Table (cascading filters)	47
A.3 Conventions & error handling	47
Appendix B - File formats (ASCII and SRF)	49
B.1 ASCII export for reflectance spectra	51
Appendix C - DB logical schema (ER + views)	53
C.1 Entity-relationship (ER) schema - generic view	55
C.2 Logical views for consultation (generic names)	55
C.3 Example read-only queries	56
C.3.1 Data Table (filtered listing)	56
C.3.2 Spectrum for plotting (ordered series)	56
C.4 Implementation notes (summary)	56
Appendix D - “Read-only” SQL examples	59
D.1 Filters (support queries for the interface)	61
D.1.1 Class (initial dropdown)	61
D.1.2 Subclass (dependent on Class)	61

INDEX

D.1.3 Sample name (dependent on Class/Subclass)	61
D.1.4 Locations (for map/filters)	61
D.2 Data Table (filtered listing + pagination)	61
D.2.1 Record list for the table	61
D.3 Record detail (panels/infobox)	62
D.3.1 Record metadata	62
D.4 Plotting (spectral series)	62
D.4.1 Primary spectrum (ordered by)	62
D.4.2 Spectra by filename (Add dataset)	62
D.5 Map (overview + carousel)	62
D.5.1 Markers by location (centroids/coordinates)	62
D.5.2 Batch for group/survey (ID list)	63
D.6 Support lookups (quick lookups)	63
D.6.1 Sample names by location	63
D.6.2 Quick support queries for the “compare” panel	63
Appendix E - UI algorithms (derivative, continuum, resampling)	65
E.1 First derivative (overlay)	67
E.2 Continuum removal	67
E.3 Resampling / SRF & user grid	67
Appendix F - UI algorithms (derivative, continuum, resampling)	69
F.1 Summary by key endpoint	71

Abstract

The INGV Spectral Library is an open-source, modular platform for the management, exploration, and analysis of spectral data across Earth Science and Cultural Heritage applications. Unlike many existing spectral libraries, which are often domain-specific and provide limited support for preliminary data interaction, the INGV Spectral Library combines structured metadata organisation, georeferenced sample exploration, and integrated spectral pre-processing tools within a single interactive environment. The platform enables users to browse heterogeneous datasets acquired from different instruments and spectral ranges, compare multiple spectral signatures simultaneously, and apply first-derivative analysis, continuum removal, and sensor-based spectral resampling through spectral response functions (SRFs). Its geospatial integration with OpenStreetMap further supports the interpretation of spectral information in relation to sample location and context. Built on a modular Frontend-Backend-Database architecture, the system supports scalable querying, consistent data retrieval, and export to external analysis environments. Its usefulness is demonstrated through geological and environmental case studies, including the kaolinite quarry at Sasso di Furbara and the salt pond of Sal'e Porcus, where the integration of in-situ measurements and bibliographic spectral resources improved mineral discrimination and process interpretation. The current roadmap includes spectral matching methods, machine-learning workflows for complex and noisy datasets, and guided ingest modules. The platform is undergoing internal validation, including checks on data consistency across database views and API outputs, usability testing of the analytical tools, and verification of query performance, prior to public release.

Keywords Spectral library; Spectral Pre-Processing; Geospatial Integration | Libreria spettrale; Pre-elaborazione spettrale; Integrazione geospaziale

Introduction

The management and analysis of spectral data are essential across several scientific and applied domains, including geophysics, geology, mineralogy, environmental studies, and cultural heritage conservation. These fields frequently rely on non-invasive techniques to determine the chemical and physical composition of materials while preserving the integrity of the investigated objects and surfaces. In this context, a spectral signature can be defined as the characteristic variation of reflectance or absorption of a material as a function of wavelength, controlled by its composition, structure, and physical state. Because different materials and surface conditions may produce distinct spectral behaviours, spectral signatures represent a key source of information for material identification, comparison, and interpretation. Surface spectroscopy has therefore become an important methodology for investigating rocks, minerals, soils, archaeological artefacts, and artworks, as it enables detailed characterisation without destructive sampling [Yogurtcu et al., 2024].

Organising spectral signatures into structured databases is important because spectral measurements are rarely useful as isolated curves alone. Their scientific value increases when they are associated with coherent metadata, such as geographic location, sample class, acquisition conditions, instrument type, and analytical context. The value of spectral data is strongly linked to information about the measurement context, which gives meaning to the data and supports their interpretation and reuse [Hueni et al., 2020]. The completeness and quality of metadata are central to enabling reliable exchange, assessment, and reuse of

field spectroscopy datasets, while helping users to evaluate the suitability of both metadata and associated spectra for a given purpose [Rasaiah et al., 2015a]. A database-oriented organisation therefore makes it possible to retrieve, compare, validate, and reuse spectra more efficiently, while also supporting reproducibility, long-term curation, and data sharing across research communities [Hueni et al., 2009]. In addition, interoperability of metadata is essential for facilitating dataset discoverability, exchange, and integration across platforms and application domains [Rasaiah et al., 2015b]. Structured spectral repositories are therefore not only archives, but also operational tools that enable systematic comparison between materials, support interpretation, and facilitate the development of advanced analytical procedures.

In the context of vegetation and agricultural applications, previous studies have shown that integrating spectral libraries with machine-learning techniques—such as k-Nearest Neighbors (KNN), Random Forest (RF), and Genetic Algorithm-Support Vector Machine (GA-SVM)—can improve the analysis of crops and phenological stages, also benefiting from spectral derivatives and specific vegetation indices [Zhang et al., 2019].

In Earth sciences, spectral data integrated with remote-sensing techniques are important not only for monitoring large areas, but also for lithological discrimination, mineralogical and alteration mapping, ore exploration, environmental geology, soil and sediment characterisation, support to geological field interpretation, and the calibration or validation of remotely sensed products [Liao et al., 2023; Peyghambari and Zhang, 2021; Wang et al., 2022]. In cultural heritage conservation, surface spectroscopy is crucial for identifying the constituent materials of artworks and assessing their state of conservation in a non-invasive manner [Ruello et al., 2022]. Although reflectance spectroscopy and other spectroscopic techniques are widely adopted, the availability of versatile and comprehensive spectral libraries for in-depth data analysis remains limited.

Existing libraries are often specialised in specific domains (e.g., rocks, minerals, synthetic materials) and do not generally provide advanced tools for preliminary interaction with the data, such as interactive comparison, integrated pre-processing, or direct exploration of spectra in relation to their spatial context (e.g., ECOSTRESS, SPECCHIO, ASTER USGS) [1; 2; 3; 4]. In addition, spectral data are often distributed across different institutions, stored in heterogeneous formats, and described with non-uniform semantic structures, which can reduce usability and make broad sharing and reuse more difficult [Xie et al., 2022].

Other widely recognised spectral libraries include the Pigments Checker spectral database, specifically designed for cultural heritage applications; the Planetary Data System (PDS) Geosciences Node [7] Spectral Library, oriented towards planetary sciences; the Arizona State University (ASU) [5; 6] Spectral Library, widely used in geology and planetary exploration; and the Infrared and Raman Users Group (IRUG) [8] Spectral Database and the Cultural Heritage Science Open Source (CHSOS) [9], primarily focused on materials characterisation in conservation science. In the cultural heritage field, dedicated open-access spectral resources remain relatively limited, despite the increasing importance of integrated databases combining XRF, FTIR, and Raman data for the identification and characterisation of art-related materials [Corteza, 2024].

While valuable and well established, these libraries generally do not offer the ability to integrate heterogeneous datasets acquired with different spectroscopic instruments, such as Raman spectroscopy, which probes molecular vibrational modes [Liao et al., 2023], Fourier-transform infrared (FTIR) spectroscopy, which identifies absorption features related to chemical bonds and functional groups [Walker, Hodgkins, Berrie, 2021], and fibre-optic reflectance spectroscopy (FORS), which measures the reflectance behaviour of surface materials in the visible-near infrared range, across multiple spectral ranges (e.g., VNIR, SWIR), thereby limiting their applicability to multidisciplinary analyses.

The library presented here has been designed with a modular and scalable architecture and provides an intuitive frontend (1) to facilitate interaction with the backend (2) and the database (3). The entire development follows an open-source paradigm.

Among existing solutions, SPECCHIO [2; 3] stands out as a particularly flexible and evolving spectral information system, combining structured data management, flexible metadata handling, and support for data ingestion, querying, and reuse across different application domains [Hueni et al., 2009; Hueni et al., 2020]. In contrast, the INGV Spectral Library directly integrates advanced analytical pre-processing tools - first-derivative analysis, continuum removal, and spectral resampling - within an interactive user interface and emphasises direct GIS integration by providing immediate capabilities for visualisation and spatial analysis in support of multidisciplinary research.

The library can ingest datasets acquired over different spectral ranges (e.g., VNIR, Visible and Near-InfraRed, and SWIR, Short-Wave InfraRed) and using multiple instruments (e.g., portable spectroradiometer, thermal camera), facilitating large-scale uploads of data collected during field campaigns, laboratory measurements, and the management of third-party data. A recent study showed that a single spectral library can be adapted to multiple sensors through data simulation (sensor spectral response functions and radiative-transfer models), enabling accurate cloud detection in multitemporal contexts [Ma et al., 2021]. Similarly, in the future the INGV Spectral Library could extend its spectra to different remote-sensing instruments, facilitating material classification and the automated removal of atmospheric effects.

This work presents the INGV Spectral Library, designed to address the existing limitations and to provide a multidisciplinary platform for the efficient management and analysis of spectral data. The main application domains include geology and Earth sciences (thematic mapping, mineralogical characterisation, support to remote sensing, and spatially explicit interpretation of surface materials) and cultural heritage conservation (non-invasive material identification, diagnosis of conservation state).

1. System Overview

1.1 Overview and architectural principles

The INGV Spectral Library is a web platform designed to guide researchers from data exploration to spectral analysis, while maintaining continuity across geographic context, metadata, and reflectance signatures. From the Home page, users select the domain of interest (e.g., mineralogy, cultural heritage, surveys) and then proceed to **Explore**, choosing whether to navigate via the map-enabling spatial access to samples-or via the filtered table-supporting targeted searches by class, subclass, location, or spectroscopy type. Opening an item in detail provides reflectance plots and analytical tools (first derivative, continuum removal, resampling/spectral response function, SRF), so that visual validation and, where relevant, export of results are immediately available.

This user flow is enabled by a modular FE-BE-DB architecture (Frontend-Backend-Database): the React frontend [10; 11] orchestrates navigation and composes parameterised requests; the Node/Express backend [16; 17] responds through consistent domain- and filter-oriented REST APIs; and the PostgreSQL database [18] exposes optimised views that unify geographic, spectral, and sample-specific metadata. The clear separation between **Map** and **Table** in the frontend, together with dedicated endpoints for lookups and spectral retrieval, improves client-side scalability (pagination, marker clustering) and clarifies data flows. A reusable component design (cards, contextual headers, tool panels) facilitates extension and maintenance, while the adoption of standard plotting and GIS libraries ensures smooth interaction even across heterogeneous datasets.

The system is open-source: code and data structures are conceived to encourage auditing, reuse, and external contributions, while maintaining a clear operational scope-consultation, basic analytical pre-processing, and export-onto which future developments can be integrated (spectral matching, machine-learning workflows, guided ingestion). In this way, the platform

combines a straightforward user journey for consulting and comparing spectra, a transparent technical pipeline for those integrating services and data, and a robust foundation for institutional curation of content and metadata.

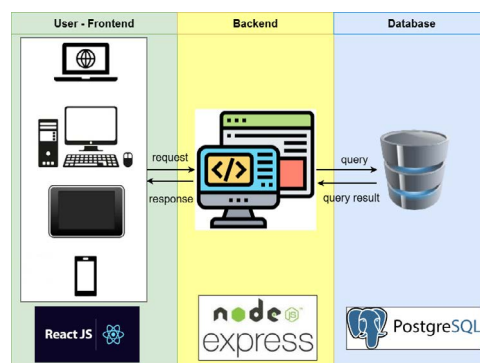
1.2 High-level diagram (Frontend $\rightleftharpoons$ Backend $\rightleftharpoons$ Database)

The platform adopts a three-tier web architecture. The React frontend (running in the browser on desktop/tablet/mobile) manages the navigation flow “Home → Explore → Map/Table → Graph” and sends HTTP requests to the Node/Express backend (API prefix /spectral/api). The backend exposes REST endpoints returning JSON (spectra, metadata, filter lists) and queries PostgreSQL, where dedicated views aggregate geographic, spectral, and sample-specific metadata. The system is stateless: each view/tool requests the required data, and the user interface renders plots and maps without maintaining server-side session state.

Arrow legend in the figure:

- **Request/response (FE $\rightleftharpoons$ BE):** GET calls with query-string parameters (e.g., type, class, id_pk) → JSON responses used to populate tables, maps, and plots.
- **Query / rows-result set (BE $\rightleftharpoons$ DB):** parameterised SQL queries against tables/views → row sets already suitable for serialization.

Figure 1 High-level architecture diagram of the INGV Spectral Library: separation between the React frontend, the Node/Express backend, and the PostgreSQL database, with request/response flows (FE $\rightleftharpoons$ BE) and query/rows flows (BE $\rightleftharpoons$ DB).



In summary, the frontend composes requests, the backend orchestrates and normalises responses, and the database provides consistent and performant result sets; this clear separation enables scalability, modularity, and independent evolution of the three layers.

2. Architecture and Data (end-to-end view: FE → BE → DB)

This chapter follows the end-to-end path of the platform from the user interface to data persistence. Section 2.1 describes the React frontend and its user workflow; Section 2.2 details the Node/Express backend and the API contract; Section 2.3 documents the PostgreSQL schema and the views used to serve maps, tables, and spectra. The goal is to clarify responsibilities and data flow across layers.

2.1 Frontend (React)

Section 2.1 is organised from “global to local”: it first describes the overall navigation flow (2.1.1), then the main pages (Home, Explore, Map/Table), and finally the Graph Dashboard

and cross-cutting components (selectors, panels, hooks, export and UI policies). This structure mirrors the actual user journey and the React routing hierarchy.

2.1.1 Global Flow (overview)

The purpose of this subsection is to provide a single narrative view of the frontend user journey before describing each page in detail.

The frontend guides users through a clear workflow: from the Home page-where the thematic domain is selected (e.g., Rock, Cultural Heritage, Surveys)-users enter the spectral data exploration interface (**Explore spectral data** view), which acts as the main navigation hub. From here, users can proceed either via the interactive map-for geographic access to samples-or via **Data Table Exploration**, which provides cascading filters and a paginated table. Both paths converge on the **Graph Dashboard UI**, the unified analysis environment where reflectance plots and tools (dataset comparison, first derivative, continuum removal, resampling/SRF) are available, and from which images and profiles can be exported. The selected context (domain, filters, record) is propagated across routes, so that the transition from overview to detail is immediate and does not rely on server-side state.

2.1.2 Home (Card-based domains)

The Home page introduces the library through a grid of thematic cards-Cultural Heritage, Rock, Minerals, Surveys, Water/Ice/Snow, Man-made, Soil, Vegetation-each accompanied by an image and a short usage description. The selection is purely exploratory and is used to set the working domain.

When a user selects a card, the app:

- stores the selected domain (source) and propagates it to subsequent routes (query string/state) to maintain consistency across views;
- redirects to the spectral data exploration interface (**Explore spectral data** view), where the pre-selected domain enables contextualised entry points for the interactive Map or the data exploration table (consistent icons, filters, and UI copy).

The page is responsive (desktop/tablet), with full-area clickable cards to reduce selection errors; short introductory text clarifies what will be available in subsequent views (maps, tables, plots, and tools). The header displays the title and institutional logo and prepares “forward” navigation without loading state on the server: each step reconstructs context from source and any parameters selected later in the workflow.

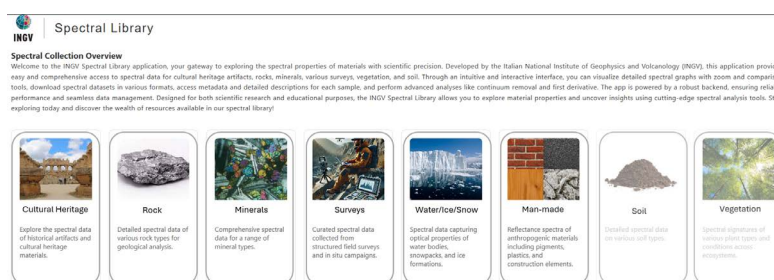


Figure 2 Home page of the INGV Spectral Library: selection of the thematic domain via cards (Cultural Heritage, Rock, Minerals, etc.), which initialises the user’s working context.

2.1.3 Explore spectral data UI (navigation hub)

After selecting a domain on the Home page, the user lands on the spectral data exploration interface (**Explore spectral data** view), a hub providing two equivalent entry points for exploration:

- **Interactive map** (/spectral/explore-data/map): geographic access to samples, with markers/clusters and an information carousel; from any point, the user can open the Graph Dashboard UI for the selected record.
- **Tabular data exploration (Data Table Exploration)**, (/spectral/explore-data/filters): targeted search through cascading filters (Class → Subclass → Sample, plus domain-specific fields) and a paginated table; each row links to the Graph Dashboard UI for the corresponding id_pk.

The page preserves the source (domain) received from Home and propagates it to subsequent views, so that labels, icons, and filters are immediately consistent. The two options are presented with equal emphasis and a short usage description, allowing users to choose the most suitable path (spatial orientation vs. table-based search) without downstream functional differences: both lead to the same analysis interface (Graph Dashboard) with an identical set of tools and export capabilities.

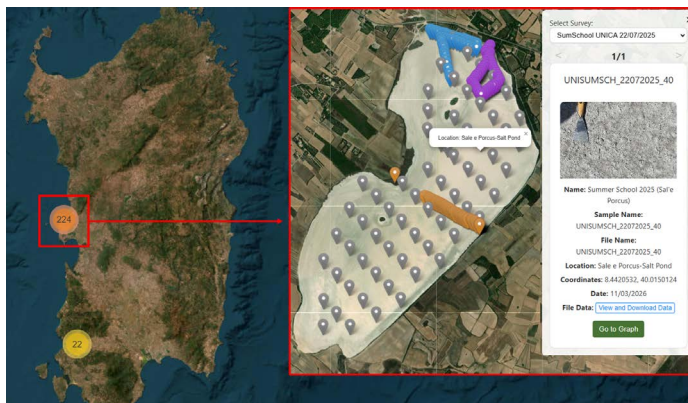
2.1.3.1 Interactive Map

The interactive map enables users to explore samples starting from their geographic context. Points are displayed using themed markers (domain-specific icons) and are grouped into clusters when density is high; zooming progressively expands clusters down to individual records. Where supported by the dataset, transects can be displayed by ordering points along a path to facilitate spatial interpretation.

On load, the page requests the minimum map payload from the backend via the data-by-location endpoint (with the type parameter derived from the selected domain). For surveys, when the user opens a group/cluster or the carousel, the UI performs a batch call to data-by-geo-ids to retrieve, in a single request, the records associated with the same survey_name. Each marker opens an information carousel showing essential metadata (sample name, location, optional image) and a direct link to the Graph Dashboard UI for that record (for the spectral-response JSON schema, see Appendix A, see Section A.2.1).

This view is intended for users who prioritise territorial orientation: it makes it possible to quickly identify spatial patterns, select an area, and move to spectral detail with a single click, while preserving domain context and navigation parameters without any server-side state.

Figure 3 “Interactive Map” view within the Explore spectral data interface: samples are displayed as geolocated markers on an OpenStreetMap basemap, with informative pop-ups and a direct link to the Graph Dashboard for spectral analysis.



2.1.3.2 Tabular data exploration (Data Table Exploration)

The tabular exploration view (Data Table Exploration) supports targeted searches through cascading filters: users start from Class and Subclass and narrow down to Sample, with additional fields appearing depending on the selected domain (e.g., Location, Survey name for surveys, Spectroscopy type, or Pigment standard for Cultural Heritage). The dropdowns are reactive: each selection updates the available options in subsequent lists, preventing empty combinations and keeping the search context consistent.

When the user runs a search, the UI sends parameterised GET requests to backend endpoints (e.g., classes, subclasses, name, locations, survey-name-by-location, spectroscopy-type, name-cult), normalising domain values where needed (for example, “minerals” → “mineral”) and composing the query string using only the active filters. Results populate a paginated table (intended for large datasets), with domain-adapted headers and a “Graph & Details” button on each row: clicking it opens the Graph Dashboard UI for the selected record (deep link via `id_pk`), where reflectance visualisation and analytical tools are available (for the table-population request, see Appendix A, see Section A.2.4 and Table A.1). This allows users oriented towards textual/semantic search to quickly identify samples of interest and move to spectral detail without changing context. The reference view underlying the tabular listing is documented in Appendix C, C.2 (entry `<domain>_datajoin_vw`; example query in C.3.1; queries for tabular listing and pagination counts: Appendix D, D.2.1).

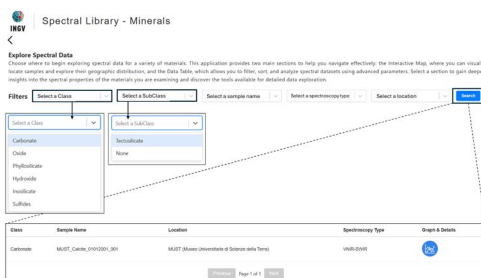


Figure 4 “Data Table Exploration” view: dynamic cascading filters and a paginated table allow rapid selection of samples by name, class, subclass, or location, with direct access to the Graph Dashboard via the “Graph & Details” button.

2.1.4 Graph Dashboard UI

The **Graph Dashboard** is the unified environment where spectral analysis and comparison take place. This section is structured by tabs (2.1.4.1-2.1.4.4) and then by cross-cutting UI components and data-access utilities (2.1.5-2.1.7), so the reader can map each tool its UI entry point and underlying API calls.

It can be reached both from the map and from the table through a deep link that includes `id_pk` and source, so the plot opens directly on the selected sample and within the correct domain (for the spectral-response JSON schema, see Appendix A, A.2.1; SQL SELECT statements for the main spectral series and “Add dataset”: Appendix D, D.4.1-D.4.2; client-side algorithms for derivative, continuum removal, and resampling: Appendix E).

On load, the dashboard retrieves the primary spectrum for the record (reflectance 0-1, X-axis in μm) along with the associated filename/metadata. The user can then work across the following tabs:

- **Reflectance (+ Add dataset):** displays the baseline spectrum and allows one or more comparison datasets to be added (by name or location). Series are overlaid with a legend/tooltips, keeping consistent units and scale to enable direct comparison.
- **Continuum removal:** computes and displays the continuum curve and the continuum-removed reflectance spectrum (**RCR**) alongside the original spectrum, in order to highlight absorption features by reducing the influence of the background spectral trend.

- **Convolution / SRF & Resampling:** performs resampling to known sensor bands (SRF) or to user-defined grids, visualising spectral response curves and producing an exportable convolved profile.
- **First derivative:** overlays the first derivative on the spectrum (toggleable Y2 axis), useful for emphasising slopes and shape changes.

The export toolbar provides two quick actions:

- **Plot Output (PNG):** exports the plot including title and institutional logo.
- **Output (ASCII.txt):** saves the numerical data (spectrum/resampled output) for external use.

The page is designed for rapid switching between tabs without reloading state: contextual parameters (domain, selected sample, any added datasets) are maintained within the navigation session, allowing users to alternate between visualisation, pre-processing, and export within a single workspace.

2.1.4.1 Reflectance (+ Add dataset)

The **Reflectance** tab displays the primary spectrum of the selected sample and allows multiple comparison datasets to be added (by name or by location), overlaying them on the same plot for direct comparison. For the mapping between endpoints and database views, see Appendix C, C.2-C.3. Operationally, on load the dashboard retrieves the record spectrum via the data-by-geo-id-spectrum endpoint; when the user adds a comparison, the UI resolves the filename and retrieves additional curves via data-by-file-name-spectrum. Spectral data in nanometres are converted to micrometres ($\text{nm} \rightarrow \mu\text{m}$) for consistency with the other tools; the Y-axis is fixed to [0-1]. In the Plotly chart [12], legend overlays and tooltips are enabled, along with PNG export and-if enabled-ASCII (.txt) export; for the ASCII format, see Appendix B, B.1 (endpoints are listed in Appendix A, A.2.1-A.2.2; example queries for record spectra and filename-based retrieval: Appendix D, D.4.1-D.4.2). This view targets users who need to validate or compare known signatures: adding series is immediate, scales are uniform, and interactive controls (zoom, hover, series toggling) allow rapid assessment of spectral similarities and differences before moving to more diagnostic tools (derivative, continuum removal, resampling).

Figure 5 Graph Dashboard UI, “Reflectance (+ Add dataset)” tab: display of the primary spectrum with multiple overlaid comparison datasets, interactive legend, and navigation controls to the other tools (derivative, continuum removal, resampling).



2.1.4.2 Continuum removal

The **Continuum** tab displays the original reflectance spectrum, the estimated continuum curve, and the continuum-removed spectrum. The latter is indicated as **RCR**, where **RCR** stands for **continuum-removed reflectance**. It represents the ratio between the original reflectance spectrum and its corresponding continuum curve and is used to emphasise absorption features by reducing the influence of the background spectral trend.

When the tab is opened, the dashboard reloads the sample spectrum using the same scale adopted in the Reflectance tab, with wavelength on the X axis, expressed in μm , and reflectance values on the Y axis, normalised between 0 and 1. The continuum curve is computed directly in the user interface.

Operationally, the spectrum is subdivided into segments and, within each segment, local maxima are identified and used as anchor points. The continuum curve is then obtained by interpolation, ensuring that it remains above the original reflectance spectrum and does not intersect it. The **RCR** spectrum is then computed by dividing the reflectance spectrum by the continuum curve. This operation flattens the spectral baseline and improves the readability of absorption-feature position and depth.

The interface overlays the reflectance spectrum, the continuum curve, and the **RCR** spectrum, with legend, tooltips, and the same zoom and inspection options available in the other spectral visualisation tabs.

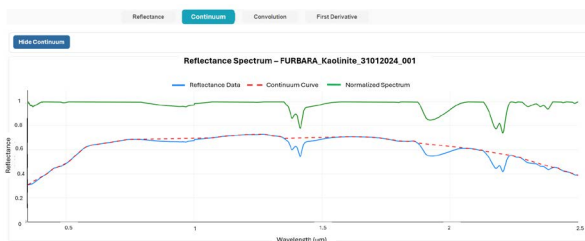


Figure 6 Example of continuum removal in the Graph Dashboard: the original spectrum is shown alongside the continuum curve and the normalised spectrum to emphasise diagnostic absorption features.

2.1.4.3 Convolution / SRF & Resampling

The Convolution tab enables spectrum resampling in two ways, either to approximate the band configuration of real sensors or to match a user-defined wavelength grid.

Sensors with known SRFs. Users select a sensor (e.g., Sentinel-2, Landsat-8, PRISMA) and the bands of interest; the interface loads the sensor spectral response functions (SRFs) and, for each band, computes resampled reflectance using weights derived from the SRF. The plot also displays the SRF curves as a visual reference, together with the resulting resampled profile. The resampled profile can be downloaded in ASCII format.

User-defined grid. Alternatively, users provide a list of target wavelengths; the UI interpolates the sample spectrum onto those values and produces a resampled profile, which can also be exported as ASCII. This is the fastest option for testing custom band sets or experimental indices. The tab operates on the same primary spectrum already loaded in the dashboard (X in μm , Y in [0-1]) and, through a dedicated selector, makes the choice between sensor/bands and a custom grid explicit. This enables rapid transitions from signature comparison to the simulation of sensor responses, supporting comparisons with remote-sensing products (examples of SRF files and data structures: Appendix B, B.2).

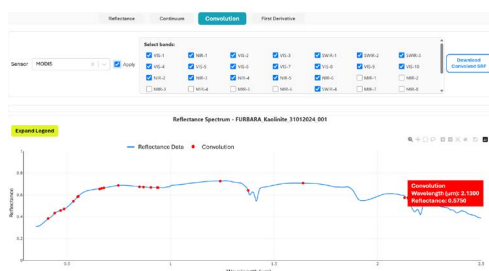


Figure 7 SRF-based resampling tool: laboratory spectrum resampled to Sentinel-2 bands, with an overlay of the spectral response curves for each band.

2.1.4.4 First derivative

The First Derivative tab adds to the reflectance curve a second series representing the local rate of change of reflectance with respect to wavelength. The goal is to highlight slopes, shape changes, and band edges that may be difficult to interpret from reflectance alone.

When activated, the dashboard computes the derivative on the same primary spectrum already loaded (X-axis in μm ; reflectance 0-1) and overlays it on the plot: reflectance remains on the primary Y-axis, while the derivative is plotted on a secondary Y-axis to keep scales separate and readable. An on/off toggle allows the derivative series to be shown or hidden in real time, and an auto-scroll brings focus directly to the plot once processing is complete, preserving user context.

Interaction remains consistent with the other tabs: zooming, hover with dedicated tooltips, and plot export from the toolbar. In combination with the other views (Reflectance and Continuum), the derivative helps confirm feature positions and delineation, especially where small differences between samples or sensors produce subtle slope variations that are hard to detect visually.

Figure 8 Continuum removal: original spectrum, continuum curve, and normalised spectrum overlaid. Normalisation highlights the relative position and depth of absorption bands.



2.1.5 Cross-cutting components (selectors & panels)

Dataset selector. From the Graph Dashboard, users can quickly add one or more comparison datasets by searching by sample name or by location. The selector calls the same lookup endpoints used by the filters (e.g., name, locations), proposes options consistent with the current domain (source), and—once a sample is selected—loads its spectrum as an overlay on the reflectance plot. Added series inherit the units and scale of the main chart (X in μm ; Y in [0-1]) and appear in the legend with compact labels (name, optional location), keeping comparisons readable even with multiple curves. Supporting lookups: Appendix A, Table A.1 (e.g., /name, /locations, /name-by-location).

Info panel. Next to the plots, an information panel summarises the key metadata of the active record (e.g., sample_name, class/subclass, location, spectroscopy type, filename). The panel updates whenever the sample changes—whether arriving from the map/table or when an additional dataset is selected—and allows the user to verify immediately that what is being displayed matches the data context. Where applicable, quick links to useful actions are also provided (open in map, show survey path, export profile).

Header & context. The header maintains a dynamic title (domain + current section) and a back button that returns to the previous view while preserving session parameters (domain, filter selections, open record). Context (source) is propagated throughout navigation—Home → Explore → Map/Table → Graph—so that labels, icons, and text remain aligned without requiring server-side state. Together, these cross-cutting components ensure continuity across the different GUIs (Graphical User Interfaces) and reduce the risk of errors when moving from overview (map/table) to analytical detail (plots and tools).

2.1.6 API client & hooks

Data access is centralised in a GET client using query-string parameters and in a set of hooks that encapsulate parameters, normalisation logic, and response formats.

Implementation excerpt (API client and hook pattern). The following code illustrates the shared GET client and the spectrum hook responsible for request composition and nm $\rightarrow$ μ m normalisation.

```

1  // src/data/api.js
2  const API_BASE_URL = `${config.URL}/spectral/api/`;
3
4  const buildQueryString = (params) => new URLSearchParams(params).toString();
5
6  export async function apiFetch(endpoint, params = {}, options = {}) {
7    const url = `${API_BASE_URL}${endpoint}${params ? `?${buildQueryString(params)}` : ''}`;
8    const res = await fetch(url, options);
9    if (!res.ok) throw new Error(`API ${endpoint}: ${res.status}`);
10   return await res.json();
11 }
12
13 // src/data/useSpectrumData.jsx (excerpt)
14 export function useSpectrumData(idPk, source, endpoint = "data-by-geo-id-spectrum") {
15   const [spectrumData, setSpectrumData] = useState([]);
16   useEffect(() => {
17     if (!idPk || !source) return;
18     const type = source === "minerals" ? "mineral" : source;
19     apiFetch(endpoint, { type, id_pk: idPk }).then(rows => {
20       setSpectrumData(rows.map(p => ({ wavelength: p.wavelength / 1000, reflectance: p.reflectance })));
21     }).catch(console.error);
22   }, [idPk, source, endpoint]);
23   return { spectrumData };
24 }

```

Figure 9 API client and data hooks: query-string construction, fetch, and nm $\rightarrow$ μ m normalisation.

Responsabilities of the shared client and hooks. The items below summarise what each hook encapsulates (parameters, normalisation, response mapping) to keep React components thin and consistent.

API client: `apiFetch(endpoint, params)`

Reads the base URL from settings (e.g., `.../spectral/api/`) and builds the query string from params, issuing a GET request that returns JSON ready for the UI. It does not set custom headers; serialization is handled by the backend.

Hook: `useFilters(source)`.

Manages cascading filters and invokes the filter aggregator endpoint, normalising source (e.g., “minerals” $\rightarrow$ “mineral”). It provides helpers to compose the filter payload and to retrieve the results table without page reloads.

Hook: `useDropdownOptions(endpoint, type, keyMapper, filters)`

Generic wrapper for dropdowns (e.g., classes, subclasses, name, locations, survey-name-by-location, spectroscopy-type, name-cult). It combines type with the current filters, calls `apiFetch(endpoint, ...)`, and maps keys via `keyMapper` before populating the options.

Hook: useSpectrumData(id_pk, source, endpoint="data-by-geo-id-spectrum")

Retrieves the primary spectrum (or, on demand, an alternative endpoint) and converts the X-axis from nm to μm for consistency with the dashboard tools; it exposes loading state and returns data already formatted for plotting.

Hook: useSRF() (SRF selector & resampling)

Handles sensor/band selection and loads the sensor spectral response functions required by the Convolution tab; it coordinates parameters used for resampling and for ASCII download of the resampled profile.

2.1.7 Export, styling, and accessibility

The interface supports immediate export of results: plots can be saved as PNG files that add the title and institutional logo only at download time (keeping the UI clean), while the resampling/SRF tools also provide ASCII export of the resampled profile or reflectance values, enabling reuse in external environments.

The visual design is desktop-first with responsive adaptations, consistent map icons, and reusable components (cards, carousel) to ensure a uniform experience across views. Attention is given to plot readability: axes and units are rendered consistently (X in μm , Y in [0-1]), series labels and tooltips are clear, and contrast supports interpretation even when multiple curves are overlaid. The map uses thematic markers and an information carousel to reduce cognitive load and to help users select the correct record before detailed analysis.

2.2 Backend (Node/Express)

This section describes the backend layer and how it exposes data and services to the frontend in a predictable, read-oriented way. The focus is on the API contract (routes, parameters, and outputs) and on how requests are routed internally to database views for maps, tables, and plots.

2.2.1 API surface and routing

The platform APIs are published under a single prefix, `/spectral/api`, so the frontend can route all requests (maps, filters, spectra) within a clear, stateless perimeter. The Express entry point mounts functional routers-`locationRoutes`, `chartRoutes`, `adminRoutes`-all under the same base path, applying CORS and JSON parsing globally; in production, the same service also serves the frontend build and falls back to `index.html` for client-side routing. A route diagnostics endpoint (`/spectral/api/_routes`) is available to quickly verify endpoint wiring during testing and validation. The API surface covers three main request families:

- **Filters and lookups:** lists of class, subclass, name, locations, survey name, spectroscopy type, with lightweight parameter normalisation (e.g., minerals $\rightarrow$ mineral) to align the UI with SQL views. These resources feed dropdowns and the tabular exploration view, without server-side state (Description aligned with backend functional documentation and FE hooks).
- **Spectral data:** retrieval of the reflectance profile for the current sample and additional datasets for comparison. The primary endpoints are `data-by-geo-id-spectrum` (spectrum of the selected record) and `data-by-file-name-spectrum` (additional curves resolved via filename/metadata). For multi-record cases (e.g., surveys), the batch endpoint `data-by-geo-ids` is also available.

- **Georeferenced data:** payloads for the map and carousel, aggregated by location and when required-by homogeneous groups (surveys). Responses are returned as JSON ready for markers, clusters, and direct links to the Graph Dashboard (See location routes).

```

1 // controllers/chartRoutes.js
2 router.get('/data-by-geo-id-spectrum', async (req, res) => {
3   let { type, id_pk } = req.query;
4   if (!id_pk) return res.status(400).json({ error: 'Missing id_pk' });
5   if (type === 'minerals') type = 'mineral';
6
7   const qMap = {
8     rock: 'SELECT wavelength, reflectance FROM rock_spectraldata_vw WHERE id_pk = $1 ORDER BY wavelength',
9     mineral: 'SELECT wavelength, reflectance FROM mineral_spectraldata_vw WHERE id_pk = $1 ORDER BY wavelength',
10    // surveys: tabella base (fix robusto)
11    surveys: 'SELECT d.wavelength, d.reflectance FROM survey_spectral d WHERE d.survey_id = $1 ORDER BY d.wavelength',
12    'man-made': 'SELECT wavelength, reflectance FROM man_made_spectraldata_vw WHERE id_pk = $1 ORDER BY wavelength',
13    wis: 'SELECT wavelength, reflectance FROM wis_spectraldata_vw WHERE id_pk = $1 ORDER BY wavelength',
14    cultural_heritage: 'SELECT wavelength, reflectance FROM cult_her_spectraldata_vw WHERE id_pk = $1 ORDER BY wavelength'
15  };
16  const sql = qMap[type];
17  if (!sql) return res.status(400).json({ error: 'Invalid type' });
18
19  try {
20    const rows = await executeQuery({ text: sql, values: [Number(id_pk)] });
21    return res.json(rows.data);
22  } catch (e) {
23    return res.status(500).json({ error: 'Internal server error' });
24  }
25 });

```

Figure 10 Express route for spectral-series retrieval: type normalisation, parameterised SELECT, and ordering by λ .

From an operational standpoint, the FE $\rightleftharpoons$ BE contract is intentionally simple: GET requests with query-string parameters (e.g., type, class, sub_class, location, id_pk) and JSON responses with homogeneous structures, so React components can render tables, maps, and plots without per-endpoint adapters. This approach-together with the single prefix and route diagnostics-reduces coupling points and facilitates both debugging and future extension of the API surface (full endpoint list with minimal parameters and response schema: Appendix A, Table A.1; call examples: A.2).

2.2.2 Request-response flow and data access

The backend receives GET requests from the frontend with query-string parameters (e.g., type, class, sub_class, location, survey_name, id_pk) and translates them into JSON responses ready for maps, tables, and plots.

Before querying the database, routers apply small semantic normalisations (e.g., minerals $\rightarrow$ mineral) so that the type of value is always consistent with the SQL views. Optional parameters are included only when present, avoiding “empty” filters. Text searches use case-insensitive patterns; for lists (e.g., batches of id_pk in surveys), the backend accepts sets of identifiers and aggregates them into a single request.

Query targets join views that are already optimised for application-level use (per domain), producing three output families:

- **Filter lists:** sequences of DISTINCT values (classes, subclasses, names, locations, spectroscopy types), alphabetically ordered and deduplicated, to populate dropdowns without client-side post-processing.

- **Spectral curves:** {wavelength, reflectance} sequences ready for plotting; the main record is retrieved via `id_pk`, while comparisons resolve filename and extract the associated curves.
- **Map payloads:** georeferenced collections including location, coordinates, and—where useful—an array of IDs to support clicks on groups/clusters and subsequent detail or batch retrieval.

Spectral datasets are returned ordered by wavelength to ensure stable plotting and expected behaviour of analytical tools (derivative, continuum removal, resampling). Reflectance is expressed in [0-1]; the frontend converts the X-axis from nm to μm consistently throughout the Graph Dashboard. Filter lists are deduplicated and cleaned of null/blank values; for the map, records can be aggregated by location or survey, reducing perceived latency when moving from overview to detail. In summary, each request is normalised, routed to the most appropriate view, and repackaged into consistent JSON, enabling the UI to move seamlessly from exploration (maps/tables) to analysis (plots/tools) while preserving a single data “grammar” throughout the workflow. Read-only `SELECT` examples for table and plot outputs are provided in Appendix C, C.3; endpoint $\leftrightarrow$ `SELECT` correspondences for filters, tables, plots, and maps: Appendix D, D.1-D.5 (complete examples for filters and spectral series: Appendix D, D.1-D.4).

2.2.3 Reliability, security, and observability

The backend is designed to provide robust and predictable services to the UI, with clear error handling, attention to input security, and minimal monitoring capabilities.

From an error-handling perspective, the APIs return distinct HTTP status codes for the main conditions: malformed requests or missing/incorrect parameters are mapped to 4xx errors, while internal issues (e.g., database exceptions, timeouts, unexpected router failures) fall under 5xx. Response payloads follow a consistent structure (status, optional message, essential details), so the frontend can display notifications and fallbacks unambiguously. During validation and testing, the route diagnostics endpoint (`/spectral/api/_routes`) helps quickly identify misconfigurations or paths that are no longer reachable.

Regarding input security, the database connection is centralised in `db.js` and configured through environment variables, avoiding the exposure of credentials in source code. Access is handled via `pg-promise`, which natively supports parameterised queries: the recommended evolution pattern is to progressively replace any SQL string interpolation with bound parameters, thereby reducing SQL injection risk and making explicit where user parameters enter the data layer. Upstream, Express applies CORS and JSON parsing globally, limiting the attack surface to legitimate requests coming from the intended frontend.

In terms of observability and performance, the service supports structured logging on the Node side (with `winston` available in the dependency set) and emits explicit messages when establishing the database connection, making configuration issues immediately visible. Application logic remains stateless and delegates the aggregation required for filters, maps, and plots to database views/tables; this facilitates the introduction of pagination techniques and result-set limits in the most critical areas (filters, large-scale lookups, surveys) and also simplifies future adoption of rate-limiting policies should the APIs be exposed publicly.

2.2.4 Configuration and deployment

The backend is configured to be portable across development and production environments, with most parameters kept outside the codebase.

Primary configuration is handled through the `.env` file, which defines variables for the database connection (host, port, database name, user, password) and, if needed, the service HTTP port. The `db.js` module reads these variables via `dotenv`, builds the connection string for `pg-promise`, and performs an initial test: the console reports whether the PostgreSQL connection succeeded, making credential or reachability issues immediately apparent.

The `app.js` entry point, in turn, reads the service port from an environment variable (with a default value if not set) and starts the Express server on that address. In development mode, the backend is typically run via `npm run server` (with `nodemon`), while “normal” execution uses `npm start`, as defined in `package.json`. The README documents the installation workflow (dependency installation, `.env` configuration, backend and frontend startup) to ensure the environment is reproducible on new machines.

In production, the same Express service can serve the frontend static assets: the build directory (e.g., `frontend/build`) is exposed as static content, and any request not recognised as an API route (`/spectral/api/...`) is redirected to `index.html`, delegating route handling to the React client-side router. This allows the application to be deployed as a single Node service exposing both the APIs and the UI, simplifying deployment on a single host or container.

2.3 Database (PostgreSQL)

The database provides persistence for samples and their spectral measurements and exposes a set of optimised views supporting the different application tools (filters, tables, information panels, plots). For the ER schema and the list of generic views, see Appendix C, C.1-C.2.

Each thematic domain (rocks, minerals, cultural heritage, surveys, etc.) follows a recurring logical pattern: a main table describing the sample, a geographic table, a domain-specific properties table, and a spectral table storing wavelength-reflectance pairs. On top of this core, derived views expose exactly the field set required by the various frontend components, minimising transformation logic in the application layer.

The schema is organised by metadata groups. Geographic metadata (locality, site descriptions, acquisition date, textual and geometric coordinates) reside in a dedicated table linked to the main table via a foreign key. Spectral metadata (spectroscopy type, any instrument classification, acquisition information) are stored in the main table and, where required, in the spectral table. Domain-specific metadata (classes/subclasses, material attributes, colours, pigment standards, manufacturer, attributions, binders, etc.) are stored in a “properties” table linked one-to-one or one-to-few with the sample. The spectral measurements themselves are stored in a dedicated table, with one row per wavelength-reflectance pair, referenced to the sample identifier.

To facilitate application-level usage, specialised views are defined on top of the “raw” tables with consistent naming across domains, for example:

- a main join view (`*_datajoin_vw`) combining sample, geographic metadata, and domain-specific properties into a single record per row, used for filters and tables;
- an extended information view (`*_datainfo_vw` or `*_panel_data_vw`) exposing a richer column set for detail panels and previews;
- a spectral-info view (`*_spectral_info_vw`) linking filename, owner, spectroscopy type, and essential metadata to geographic context;
- a spectral-data view (`*_spectraldata_vw`) returning wavelength-reflectance pairs directly, already associated with the sample identifier and key sample fields, ready to be queried by plot-related endpoints.

Key columns and view usage are summarised in Appendix C, C.2; reference read-only SELECT statements for consultation are provided in Appendix D, D.1-D.5.

For survey use cases, a dedicated spectral table is available (e.g., `survey_spectral`), designed to handle sets of repeated measurements and to support batch endpoints returning multiple spectra in a single call. Downstream, all result sets can be readily transformed into ASCII formats (resampled spectra, SRF profiles, sample tables), which the Graph Dashboard makes exportable for external use.

PostgreSQL is therefore responsible both for referential consistency across samples, metadata, and spectra, and for query performance, through the use of primary/foreign keys, indexes, and views designed around the platform's real usage scenarios. Backend APIs rely on these views through parameterised queries, obtaining row sets already structured for the different interface tools (filters, maps, plots) and reducing application-code complexity

2.4 GIS integration and map (OpenStreetMap)

The Map component natively integrates the spectral library with a geographic context, enabling users to explore samples directly on the map and to reach spectral detail in a few steps.

Interoperability between the spectral database and a GIS environment is crucial to fully exploit the spectral library and its consolidated dataset [Cardoso-Fernandes et al., 2023].

On load, the map requests from the backend the minimum payload required via the `data-by-location` endpoint, parameterised by the selected domain (type/source). The response contains, for each locality, coordinates and the list of associated identifiers; these are rendered as markers on an OpenStreetMap basemap [14; 15]. Where density is high, markers are clustered to avoid overlap; progressive zooming expands clusters down to individual samples.

Marker styling is managed centrally through Location Icons, which maps each domain (or category) to a set of thematic icons consistent with the library legend (rocks, cultural heritage, surveys, etc.).

Where a locality includes multiple samples or repeated measurements, clicking a marker opens a carousel: a series of cards showing essential metadata (sample name, locality, optional image, brief summary information) and a button to access the detail view. For survey-type datasets, when needed the map can request a set of samples in a single call via the `data-by-geo-ids` endpoint, enabling rapid population of the carousel and-where implemented-the rendering of transects by ordering points along an interpretive path.

From each carousel card, the user can open the Graph Dashboard for the selected record via a deep link that includes the sample identifier and the current domain. In this way, the map is not merely a background view but a true entry point: users can start from territorial context, identify an area or locality of interest, and reach reflectance plots and analytical tools in a few clicks, while preserving the link to location and to the sample group from which they started. Endpoints used: Appendix A, A.2.3 (map, batch) and Table A.1; queries for markers/clusters and map-driven batch retrieval: Appendix D, D.5.1-D.5.2.

2.5 Security and traceability (brief notes)

The platform is designed primarily for internal, read-oriented use, accessible via a browser only to users who hold credentials on a dedicated machine and is not publicly exposed. Data ingestion is handled through controlled workflows: spectra and metadata are uploaded and validated in separate contexts, while the web interface exposes only search, visualisation, and export functionality. In this way, users accessing the library for analysis do not have direct access to destructive or modification operations on the operational database.

From a technical standpoint, baseline security relies on several structural elements. The database connection is configured via environment variables (the `.env` file), so sensitive parameters

such as host, port, database name, and credentials do not appear in the source code and can be differentiated between development and production. The Express application applies a controlled CORS configuration and centralised request parsing, delimiting the perimeter of legitimate API calls. At the application layer, the use of pg-promise and a single DB access layer supports (and makes advisable) the systematic adoption of parameterised queries, avoiding direct SQL string interpolation and reducing injection risk.

Traceability and observability are supported through structured logging based on winston and explicit startup messages (e.g., for verifying DB connectivity). Application and database errors can be logged with appropriate severity and context, enabling maintainers to quickly reconstruct the request-response cycle in the event of malfunctions. If, in the future, API access is extended beyond internal use, the current design readily supports the introduction of rate limiting, more granular DB-side roles and permissions, and stricter validation of incoming parameters, while keeping responsibilities separated: configuration and hardening managed by INGV maintainers, input control and normalisation handled by the backend, and permission enforcement and referential integrity guaranteed by PostgreSQL. Conventions for query strings, formats, and API errors: Appendix A, A.3.

3. Spectral analysis tools (GUI)

The INGV Spectral Library provides a set of tools designed to support the transition from simple inspection of a spectral signature to operational interpretation. In reflectance spectroscopy, each material is described by a curve expressing the fraction of reflected energy as a function of wavelength; the overall shape, slopes, and absorption bands depend on composition, physical state, and measurement conditions. For this reason, the first analytical step is always the visualisation and direct comparison of one or more curves on a consistent scale, verifying that spectra are comparable in terms of domain, spectroscopy type, and spectral range.

Dynamic filters and the data exploration table help build a well-justified comparison set: selecting by class/subclass, spectroscopy type, location, or survey prevents inappropriate comparisons and isolates homogeneous groups for which analytical tools have physical meaning. Once the samples of interest have been identified, the Graph Dashboard offers multiple levels of analysis. Reflectance visualisation captures the overall trend and primary differences between curves; the first derivative emphasises local variations, highlighting slope changes and band edges; continuum removal reduces baseline effects, bringing the position and relative depth of diagnostic features into focus.

When measurements need to be related to the remote-sensing scale, the resampling and SRF tool enables laboratory signatures to be adapted to real sensor bands or user-defined grids, bringing the reflectance curve closer to the spectral products with which it will be compared. Taken together, these tools support a shift from a purely qualitative reading of curves to a more structured analysis, while maintaining continuous visual control of what happens to the spectrum at each stage.

3.1 Real-time visualisation and simultaneous comparison of spectral signatures

As a first level of analysis, the Graph Dashboard enables real-time visualisation of a sample spectrum and simultaneous comparison with other spectra selected from the library. All operations start from reflectance data stored in the database (wavelength-reflectance pairs) and served through dedicated endpoints; the frontend normalises the X-axis to μm and the Y-axis to [0-1].

Once a sample is opened from the map or table, the **Reflectance** tab automatically loads

the primary spectrum and plots it in an interactive chart. Users can then add one or more comparison datasets (for example, samples within the same class, samples from a different location, or bibliographic reference signatures) by selecting them from lists filtered by name or location. Each new series is overlaid on the same axes, with dedicated legend entries and tooltips, while preserving the reflectance scale and wavelength units.

Operationally, this makes it possible to:

- quickly verify consistency across spectra (same overall shape, same main absorption bands);
- highlight differences due to compositional variability or acquisition conditions;
- prepare for more targeted analyses: if “by-eye” comparison suggests similarities or discrepancies, users can switch to the **First derivative**, **Continuum removal**, or **Convolution/SRF** tabs to investigate specific portions of the curve.

Plot updates are immediate: any addition or removal of a comparison dataset is reflected in real time, with zoom and pan available to focus on specific spectral intervals (e.g., a restricted VNIR or SWIR window). The export toolbar allows the chart to be saved as a PNG including title and logo, enabling the comparison to be reused directly in reports, presentations, or technical documentation without intermediate steps. For loading the initial spectrum, see Appendix A, A.2.1; related lookups/filters are listed in Table A.1.

3.2 Dynamic cascading filters and reactive table

Dynamic cascading filters are designed to build a coherent subset of samples before users proceed to spectral comparison. Rather than working with long, weakly structured lists, selection is performed in successive steps:

- **Class:** macro-category (e.g., lithotype, material family, object type).
- **Subclass:** specialisation within the selected class.
- **Sample:** individual sample within the chosen Class/Subclass combination.

In addition to these three cross-domain levels, domain-specific fields are available—for example **Location** and **Survey name** for survey-type datasets, **Spectroscopy type** to distinguish instruments/measurement modes, and **Pigment standard** for cultural heritage. Dropdown menus are reactive: each choice filters the next step, preventing combinations that do not exist in the database and reducing the risk of “out-of-context” comparisons.

When the user runs a search, the selected values are translated into a parameterised request to the backend, which queries the join views and returns a record list that already satisfies the imposed constraints (domain, classes, locations, spectroscopy type, etc.). The resulting reactive table is paginated and domain-adapted (dedicated columns for pigments, lithologies, manufacturers, etc.), and includes, for each row, a direct link to the Graph Dashboard for the selected sample.

In this way, users never reach the analysis tools “blind”: the curves they will visualise and compare come from a sample set that is homogeneous with respect to classification and key metadata, constructed through cascading filters and easily reviewed in the table before moving to spectral detail.

3.3 First Derivative

The first derivative of a reflectance spectrum measures how reflectance changes from one wavelength to the next. In practice, instead of focusing only on the curve’s “absolute” values,

it describes how much-and in which direction-the curve rises or falls: positive values indicate increasing segments, negative values indicate decreasing segments, and values close to zero correspond to nearly flat portions. This perspective is useful for highlighting band edges, slope changes, and subtle variations that may be difficult to perceive in reflectance alone.

In the system, the derivative is computed numerically from the spectrum already ordered by wavelength. For each pair of adjacent points, the change in reflectance is estimated relative to the change in, producing a sequence of derivative values associated with the same μm wavelength scale used in the original plot. The computation is lightweight and designed for interactive use: it relies on the actual sampled points available (i.e., it does not introduce additional interpolation) and, for this reason, its behaviour is influenced by point density and by any noise present in the raw data.

In the Graph Dashboard, the derivative is displayed as an overlay on the reflectance curve: the original spectrum remains on the primary Y-axis, while the derivative series is plotted on a secondary Y-axis to keep both scales readable. A dedicated toggle allows users to enable or hide the derivative in real time, enabling a quick switch from a “reflectance-only” view to a combined “reflectance + local variations” view. When the derivative is enabled, the interface moves focus to the plot to ensure the computed result does not go unnoticed. In combination with continuum removal and multi-spectrum comparison, this tool helps confirm the position and shape of the most informative portions of the curve, while always preserving the original spectral signature as reference (operational details and formula: Appendix E, E.1).

```

1  // src/Pages/Graph/utils/calculateDerivative.js
2  export function calculateDerivative(series) {
3    const out = [];
4    for (let i = 0; i < series.length - 1; i++) {
5      const dx = series[i+1].wavelength - series[i].wavelength; //  $\mu\text{m}$ 
6      if (dx <= 0) continue;
7      const dy = series[i+1].reflectance - series[i].reflectance;
8      out.push({ wavelength: series[i].wavelength, reflectance: dy / dx });
9    }
10   return out;
11  }

```

Figure 11 Computation of on adjacent points for derivative overlay.

3.4 Continuum Removal

Continuum removal is a technique designed to separate a spectrum’s baseline trend from its absorption bands. In a real reflectance spectrum, the curve is not composed only of narrow minima and maxima: it is superimposed on a slowly varying background with wavelength (instrument effects, measurement geometry, and the material’s general optical properties). This background can “mask” the true depth of absorption features. The idea of continuum removal is to construct an upper envelope of the spectrum (the continuum) and then normalise the original curve against this envelope, so that absorption bands stand out as depressions on a flattened baseline.

In the system, continuum computation is performed entirely on the frontend starting from the spectrum returned by the backend. The dedicated spectral endpoint provides

wavelength-reflectance pairs ordered by; the Graph Dashboard converts the X-axis to μm and passes these data to the continuum-computation module.

The adopted algorithm follows a segment-and-peak approach:

- the spectrum is divided into a given number of intervals along the wavelength axis, covering the full range with a set of windows;
- within each interval, one or more peaks (local maxima) are identified and treated as continuum anchor points;
- these points are then connected via interpolation to build an envelope curve that, by construction, remains above the original reflectance or touches it without dropping below.

Once the continuum is obtained, the spectrum is normalised point by point by dividing the observed reflectance by the continuum value at the same wavelength. The result is a normalised spectrum in which the baseline trend is suppressed, and absorption bands appear as sharper depressions, with depths and widths that are more comparable across different samples under the same domain and acquisition conditions.

```

1 // src/Pages/Graph/Utils/continuumUtils.js
2 export function calculateContinuumAndNormalization(series) {
3   const nodes = pickEnvelopeNodes(series); // robust heuristics on local maxima
4   const contF = interpolateSegments(nodes); // linear interpolation above R(λ)
5   const continuum = series.map(p => ({ wavelength: p.wavelength, reflectance: contF(p.wavelength) }));
6   const normalized = series.map(p => ({ wavelength: p.wavelength, reflectance: p.reflectance / contF(p.wavelength) }));
7   return { continuum, normalized };
8 }

```

Figure 12 Continuum envelope and normalised spectrum for diagnostic overlay.

The BE→FE flow is intentionally simple: the backend provides only raw spectral data (ordered by, reflectance in [0-1]); all logic for peak detection, continuum interpolation, and normalisation is implemented in the **Continuum** tab of the Graph Dashboard. The final visualisation overlays the original spectrum, the continuum curve, and the normalised spectrum: users can enable or disable each series, observe how features change when moving from “raw” reflectance to the normalised spectrum, and use the same export toolbar as the other tools to include results in plots or tabular form (steps, normalisation, and limitations: Appendix E, E.2.).

3.5 Resampling & SRF

The resampling & SRF tool aims to bring a laboratory spectrum closer to the spectral sampling of real sensors or to user-defined grids. In practice, instead of working with all wavelengths measured by a field/laboratory instrument, the system reconstructs a “condensed” version of the spectrum, sampled only where a real sensor is sensitive or where the user chooses to focus attention. This is particularly useful when comparing reference spectral signatures with remote-sensing products or when defining custom bands/indices.

This approach is supported by studies showing that sensor-specific variations in the relative spectral response (RSR) can lead to significant differences in measured reflectance and NDVI [Trishchenko et al., 2002]. In addition, accurate in-flight calibration of the RSR is critical to ensure radiometric consistency in imaging spectroscopy data [Cogliati et al., 2021].

Within the Graph Dashboard, the tool is organised into two alternative modes.

Mode A-Sensor selection with known SRFs. In the first mode, the user selects a sensor from a predefined list (e.g., Sentinel-2, Landsat-8, PRISMA) and chooses the bands of interest. For each band, the system loads the corresponding sensor spectral response function (SRF), i.e., the curve describing how that band weights different wavelengths. Starting from the original spectrum already loaded in the dashboard, the tool computes a band-equivalent reflectance value by combining the available measurements using weights derived from the SRF. The result is a resampled profile that approximates the sensor response for the sample under analysis.

In the plot, SRF curves are also displayed alongside the reflectance curve as support functions/band overlays, making it immediately clear which portions of the spectrum contribute to each resampled value. This band overlay helps interpret differences between sensors (narrower vs. broader bands, slightly shifted band positions) and clarifies which spectral features effectively contribute to each band measurement. The resampled profile can be exported as ASCII, enabling reuse in subsequent analyses or direct comparison with products derived from the real sensor.

Mode B-Custom files and user-defined grids. In the second mode, the user does not start from a real sensor but from a custom spectral grid. The user can provide a list of target wavelengths (e.g., band centres of an experimental instrument, or positions designed for a new index), and the Graph Dashboard interpolates the sample spectrum at those values, producing a new resampled profile. In this case as well, the result can be exported in ASCII format and can be overlaid on the continuous spectrum to assess how much information the selected grid captures-or loses-relative to the original curve.

```

1  // src/Pages/Graph/Tabs/useSRF.js
2  export function applySRF(series, srf) {
3    const convolvedSpectrum = srf.bands.map(({ band, srf: pairs }) => {
4      let num = 0, den = 0;
5      for (const [lam, w] of pairs) {
6        const r = interpLinear(series, lam); // μm -> reflectance
7        num += r * w; den += w;
8      }
9      return { x: band, y: den ? num / den : null, band }; // x = λ band centrale/indice
10   });
11   return { convolvedSpectrum, srfDatasets: [] }; // optional: SRF curves for overlay
12 }
13
14 // user-grid fallback
15 export function resampleOnGrid(series, grid_um) {
16   return grid_um.map(lam => ({ x: lam, y: interpLinear(series, lam) }));
17 }

```

Figure 13 Resampling: SRF-weighted average and alternative resampling on a user-defined grid.

In both modes, the data flow is the same: the backend provides the raw spectrum (ordered by, reflectance in [0-1]) and, where required, sample-associated information; resampling is performed entirely on the frontend, on the same μm wavelength scale used by the other tools. The resampled spectrum and SRF bands are then displayed together with the original spectrum, with the ability to enable/disable individual series and export results. This enables a transparent transition from the “fine” laboratory spectral signature to a representation compatible with the sensors or operational grids used in remote-sensing or modelling workflows (examples of SRF

files and data structure: Appendix B, B.2; SRF-based convolution and user-grid resampling: Appendix E, E.3; SRF formats: Appendix B, B.2).

4. Ingest & Data Pipeline

This section explains how spectral files and metadata are ingested, validated, and normalised before being stored in the database and exposed to the UI via dedicated views.

4.1 Ingest workflow: spectra and metadata

Data ingestion into the library is designed as a controlled workflow in which spectral files and associated metadata are collected, checked, and then inserted into the structured tables (sample, geography, domain-specific properties, spectral data) that feed the platform's views and tools. In practice, ingestion is performed through Python scripts operating on one or more input directories: spectral files (e.g., ASCII or CSV files containing wavelength-reflectance pairs) and, where available, supporting metadata files (field campaign sheets, laboratory forms, structured descriptions). The script parses the files, extracts key information (sample name, domain, class/subclass, spectroscopy type, and any references to location or survey), and maps it to the fields defined in the application tables.

During this stage, a set of minimal validations is applied to ensure that the data entering the database are at least consistent and usable by the analytical tools:

- checking the presence of mandatory fields (sample identifier, domain, measurement type, non-empty spectral series);
- basic verification of the spectral curve (numeric and monotonic wavelengths, reflectance values within an expected range, absence of clearly malformed rows);
- consistency between metadata and spectral files (e.g., matching between the filename and the identifier reported in forms/sheets).

In parallel, the script applies basic normalisation steps to keep the archive homogeneous downstream: it standardises domain and class names, harmonises measurement units where needed, ensures wavelengths are stored in a single consistent format, and enforces coherent conventions for the textual fields used by filters (class, subclass, location, pigment standard, etc.). After these checks, data are inserted or updated in the “raw” tables (samples, geography, properties, spectra), while derived views (*_datajoin_vw, *_spectral_info_vw, *_spectraldata_vw) expose records in forms already suitable for filters, maps, and plots.

The outcome is a workflow in which potentially critical operations (file parsing, validation, normalisation) occur upstream at ingest time, while the online component of the platform always operates on pre-validated data-reducing both the risk of consultation-time errors and the need for corrective logic within individual endpoints or GUIs.

4.2 Supported formats and naming conventions

The library intentionally relies on simple, easily reusable formats-both for ingest and export-to facilitate integration with other tools and working environments.

For spectra, the reference exchange format is a tabular ASCII file (typically .txt or .csv) with at least two columns: wavelength and reflectance. During export from the Graph Dashboard, files produced by the tools (original reflectance, SRF-resampled spectra, custom-grid

outputs) follow this minimal structure, so they can be reopened without friction in external environments (Python, R, spreadsheets) and reused in subsequent pipelines. Inputs for some advanced features—such as user-defined spectral grids in the resampling tab—also rely on simple ASCII listings (a list of target wavelengths), so users can prepare them with any editor or script.

Sensor spectral response functions (SRFs) are handled through structured files that associate each band with its response curve as a function of wavelength. Depending on the use case, these can be encoded as JSON or in tabular form, but the underlying logic is the same: a set of wavelength-weight pairs per band, which the convolution tool uses to compute resampled values and to render band overlays on the reflectance plot.

In terms of nomenclature, the platform adopts conventions designed to ensure stable references and consistency across metadata, files, and views:

- each sample is associated with a domain-unique filename, which acts as the bridge between ingest files, database tables, and spectral views;
- domain and class labels (e.g., variants such as “minerals” vs. “mineral”) are normalised during ingest and by the backend, so endpoints and filters always operate on consistent labels;
- user-facing fields (class, subclass, location, pigment standard, etc.) follow homogeneous textual conventions, supporting both manual search and future automated use in analysis algorithms.

This combination of lightweight formats and stable naming keeps the library open to external reuse (export and integration) without introducing proprietary formats, while preserving the precise linkage between original files, database records, and GUI analysis tools (real examples of ASCII export and SRF structures: Appendix B, B.1-B.2).

5. Validation, performance, and reliability

This section describes the validation approach used to ensure consistency between database contents, API responses, and the behaviour of the GUI tools, with checks spanning data integrity, endpoint fidelity, and user-facing functionality/performance.

5.1 Testing methodology

Validation of the INGV Spectral Library is based on a combination of data checks, endpoint verification, and functional testing of the GUI tools, with the goal of ensuring that what users see in the Graph Dashboard effectively matches what is stored in the database and aggregated through the views.

Consistency across raw data, views, and GUI

The first testing layer addresses internal consistency of the data model. For each domain, a set of “reference” samples is selected and the following is verified:

- records present in the base tables (samples, geography, properties, spectra) are correctly represented in the join views and spectral views (`*_datajoin_vw`, `*_datainfo_vw`, `*_spectral_info_vw`, `*_spectraldata_vw`);
- linking keys (sample identifier, filename, and any survey ID) preserve the same meaning throughout the chain, enabling unambiguous navigation from a record to a spectrum and vice versa;

- values critical for consultation (class, subclass, location, spectroscopy type, pigment standard, etc.) remain consistent between tables and views and are not subject to unintended transformations during joins.

For these reference samples, direct PostgreSQL queries are executed, and results are compared with what is delivered to the frontend via the REST endpoints, ensuring there are no discrepancies in content or ordering (in particular, wavelength ordering in spectra).

Accuracy of views and endpoints

The second testing layer focuses on endpoint fidelity with respect to database views and UI expectations. For each service family, a small set of test cases is defined:

- for cascading filters, verifying that endpoints return deduplicated, ordered lists consistent with the imposed constraints (Class → Subclass → Sample, location, survey name, spectroscopy type, etc.), and that dropdown options correspond to values present in the underlying views;
- for spectra, comparing curves returned by the dedicated endpoints (spectrum by id_pk, spectrum by filename, batch spectra) against rows in the spectral views, verifying that wavelength ordering is preserved and that reflectance values are unchanged relative to the database;
- for the map, checking that coordinates, localities, and the identifier lists served by the geospatial endpoint can effectively reconstruct the markers and sample groups displayed in Map.js and in the associated carousel.

These tests can be executed manually (by comparing JSON exports and SQL queries) or semi-automatically, using scripts that query endpoints directly and verify correspondence with database queries.

Functional and UX testing of analysis tools

The third testing layer targets spectral analysis tools and their behaviour in the GUI. Using a selected set of spectra (e.g., simple curves with well-known bands and more complex cases), it is verified that:

- the Reflectance tab displays the primary spectrum and any added datasets on consistent axes (X in μm , Y in [0-1]), and that series added via selectors correspond to the samples actually chosen by the user;
- the first derivative correctly highlights slope variations (e.g., remaining regular for monotonic spectra, and showing peaks and sign changes at band edges for spectra with clearly defined absorption features);
- continuum removal produces an envelope curve that remains above the original spectrum and a normalised spectrum in which absorption bands are effectively more readable, without introducing evident artefacts;
- the resampling & SRF tool generates profiles consistent with the provided SRFs or user-defined grids, and that exported ASCII files match the values displayed in the plot.

These checks also include UX aspects: presence and clarity of messages when errors occur or data are missing, interface responsiveness when switching samples or tools, and correct updates of information panels and legends. In this way, the testing methodology covers not only numerical correctness of transformations, but also readability and perceived reliability for users working with the tools in day-to-day practice. The list of measured endpoints is reported in Appendix A, Table A.1; response schemas used in the tests are in A.2; files used for verification (ASCII and SRF) are in Appendix B, B.1-B.2; a numerical summary of measurements by scenario/endpoint is provided in Appendix F.

5.2 Query performance and average endpoint response times; DB/BE scalability notes

Platform performance was evaluated by measuring REST endpoint response times across three representative scenarios on an internal network:

- exploration via the map (Interactive Map),
- consultation via the Data Table with cascading filters,
- use of the Graph Dashboard with analysis tools (Add dataset, Continuum, Resampling/SRF, First derivative).

Measurements collected via browser DevTools indicate that:

- lookup and filter endpoints (classes, subclasses, sample names, name-by-location) respond within a few milliseconds and are effectively transparent to the user;
- endpoints that populate the table or load the map return results within a few hundred milliseconds, with heavier cases still within one second;
- retrieval of a sample's primary spectrum typically ranges from 0.13 s (minerals) to 0.5 s (survey from map), while additional comparison spectra are much lighter (~50 ms);
- once the primary spectrum and any additional datasets are loaded, derivative, continuum removal, and resampling/SRF operate entirely on the frontend using in-memory data, without generating significant additional backend traffic, keeping the Graph Dashboard responsive during subsequent interactions.

Table 1 summarises the observed order of magnitude of the response times for the main API endpoints tested during the validation sessions.

Scenario	Endpoint	Primary role	No. of measurements	Average time (ms)	Approx. range (ms)
Data Table (minerals)	/classes?type=...	Class list	1	~90	~90
Data Table (minerals)	/subclasses?...	Subclass list	2	~9	8-9
Data Table (minerals)	/name?...	Sample name list	2	~8	7-8
Data Table (minerals)	/filter?...	Results-table population	2	~406	18-793
Map (survey)	/data-by-location?type=...	Map marker/cluster	2	~310	305-316
Map (survey)	/data-by-geo-ids?...	Survey sample batch retrieval	2	~135	129-140
Info/Detail	/data-by-geo-id?...	Sample metadata (table, map, graph)	8-47	~80-100	15-311
Graph Dashboard	/data-by-geo-id-spectrum?...	Primary spectrum	2+8	~130 / ~236	118-304
Graph Dashboard	/data-by-file-name-spectrum?...	Additional spectra (Add dataset)	3	~49	37-67

Graph Dashboard	/locations?...	Contextual location lookup	1	~157	~157
Graph Dashboard	/name-by-location?...	Sample names by location	2	~14	13-14

Table 1 Summary of the observed response times for the main API endpoints tested during the validation sessions. Values are reported as mean response times and indicative ranges across repeated measurements, and are intended to document the order of magnitude of backend performance under the tested operating conditions.

For endpoint $\leftrightarrow$ payload correspondence, see Appendix A, Table A.1 and A.2; the official snapshot with N/min/avg/max by endpoint is provided in Appendix F. Overall, the user experience is consistent with an interactive system: loading the filtered table and the map requires a few tenths of a second, opening a single spectrum in the Graph Dashboard typically falls between 0.2 and 0.5 seconds depending on the domain, and subsequent operations (tool activation, zoom, curve overlays) do not introduce additional backend latency.

From a scalability perspective, the endpoints most sensitive to dataset growth are those returning large numbers of records (table, map, survey batch) and those operating on particularly dense spectra. However, the use of dedicated views, table pagination, marker clustering in the map, and a stateless backend design provide headroom to handle larger volumes, with targeted options available (additional indexes, “slim” views, and potential caching) should response times increase as the library expands.

5.3 Current limitations

Although the platform is already operational for consultation and basic analytical pre-processing, a number of structural and functional limitations are intentionally accepted at the current stage. A first constraint concerns data ingest: a user-facing upload workflow via the web interface is not yet available. The insertion or update of spectra and metadata is performed through scripts and controlled procedures managed by the library curators. This approach ensures data consistency and quality, but it limits spontaneous contributions and slows the integration of new external datasets.

From an analytical-content perspective, the integrated tools (first derivative, continuum removal, resampling/SRF) operate directly on stored data, without automatic denoising or smoothing steps. For particularly noisy spectra, or for highly irregular sampling, numerical derivatives and continuum estimation may amplify noise or produce features that are difficult to interpret; these tools therefore assume informed use by users familiar with the quality and provenance of the input data.

The resampling & SRF component also reflects an early-stage implementation: support is centred on a subset of sensors and on the available SRFs. Users can still define custom grids, but systematic coverage of all operational sensors and spectral ranges is not yet provided. In addition, transforming high-resolution spectra into resampled profiles inevitably involves a loss of fine detail, which must be considered when interpreting narrow bands or complex structures. At the infrastructure level, the library is designed for internal use and does not yet expose advanced data versioning, application-role management, or fine-grained authentication. The database stores the current version of samples and spectra without a change history, and the backend does not currently implement rate limiting or API keys for public use; these features are

envisaged as potential extensions in view of future external access and more detailed auditing. Finally, the platform is undergoing internal validation in terms of performance and load. Scalability to larger datasets is supported through dedicated views, pagination, marker clustering in the map, and a stateless backend design, but quantitative benchmarks (average endpoint times, behaviour under load) are still being defined and will be documented in a later phase, together with any targeted optimisations.

6. Deployment and runtime environment

This section summarises how the platform is deployed and configured as a three-tier stack, outlining the minimum runtime requirements for PostgreSQL, the Node/Express API service, and the React frontend, with configuration handled via environment variables (.env) and a frontend config file for the API base URL.

6.1 Runtime requirements, environment variables, and configuration

The platform is designed to run as a three-tier stack (DB, backend, frontend), with modest requirements and an environment-variable-based configuration.

Database

The data layer requires a PostgreSQL instance reachable by the backend, with the application schema (tables for samples, geography, properties, and spectra) and the corresponding views (*_datajoin_vw, *_datainfo_vw, *_spectral_info_vw, *_spectraldata_vw, etc.) already created. Connectivity is configured through environment variables read from .env, which include hostname, port, database name, and credentials for the role used by the application service.

Backend (Node/Express)

The backend requires a Node.js environment compatible with the version specified in package.json (20.x series) and dependency installation via npm install. The Express application listens on a configurable port (default 5001 or the value set in PORT) and mounts routers under the /spectral/api prefix. The database connection is initialised at startup via pg-promise, using parameters loaded through dotenv. The main scripts are npm start for standard execution and npm run server for development with automatic reload.

Minimum environment variables include:

- database host, port, name, and credentials;
- the optional HTTP service PORT;
- additional parameters for logging and mode (development/production), if required by the infrastructure.

Frontend (React)

The frontend is a React application built with a standard toolchain (react-scripts) and a set of libraries for routing, plotting, and mapping (React Router, Plotly, Leaflet/MapLibre, etc.) [13]. In development it runs via the usual npm start on its dev server; in production it is compiled with npm run build and served as static content by the same Express backend, which falls back to index.html for client-side routes.

Endpoint configuration is handled via a frontend configuration file (e.g., config.json), which defines the API base URL (typically `http://<host>:5001/spectral/api/` in a local environment) and can be adjusted to target different hosts/ports in production.

Operational summary

In summary, deployment requires:

- a PostgreSQL instance with the library schema and views;
- a Node/Express service configured via `.env`, with database connectivity verified at startup;
- a built React frontend served by the same backend (or by a front-facing reverse proxy), with paths to `/spectral/api` aligned to the network configuration.

This setup allows the platform to run on a single host or container, while preserving logical separation between DB, API, and UI and simplifying migration across development, testing, and production environments.

6.2 Repository structure, startup scripts, logging & monitoring

The INGV Spectral Library is organised as a single application repository, with the Node/Express backend at the root and the React frontend in a dedicated subfolder. The backend includes the `app.js` entry point, controllers (`controllers/`), database-access modules (`utils/db.js`, `dbHandler.js`), and configuration files (`package.json`, `.env`, `README.md`). The frontend lives under `frontend/`, with its own `package.json`, the `src/` structure (components, pages, hooks), and a `config.json` file for endpoint configuration. In production, the frontend build is generated under `./frontend/build` and is served directly by Express.

Startup scripts follow a clear dev/prod separation. For the backend, `package.json` exposes `npm start` (execution via `node app.js`) and `npm run server` for development with `nodemon`, enabling automatic reload when files change.

For the frontend, its `package.json` defines `npm start` for development mode (`react-scripts start`) and `npm run build` for compilation; at the end of the build, a `move-files` script (`node moveFiles.js`) moves artefacts into the location expected by the backend, so that `app.js` can serve them as static content. The project `README` summarises the installation and startup workflow in development, explicitly indicating the Node version to use.

On logging and monitoring, the backend adopts a combination of console logging and structured logging. At service startup, an explicit message always reports the listening port, making it easy to confirm that the process is running.

The database connection module (`db.js`) uses `dotenv` and `pg-promise` to initialise the PostgreSQL connection and logs the outcome of an initial connectivity test, making credential or reachability issues immediately visible. The backend dependency set also includes `winston`, which-when configured-enables structured logs and forwarding to files or external monitoring systems.

For application diagnostics, a dedicated endpoint is available at `/spectral/api/_routes`, returning the list of active routes registered in Express. This facilitates API-surface checks during testing, helps quickly identify unreachable paths, and can be queried by simple health/monitoring checks. Combined with database logs and Node/Winston logs, this setup provides a sufficient monitoring baseline in an internal environment; integration with external tools (process managers, latency metrics, alerting) can be added without changing repository structure or startup scripts.

6.3 Containerisation and CI roadmap

In its current configuration, the platform is intended to run on a single dedicated machine, hosted at a remote site and managed internally. Access is provided via a browser behind credential-based authentication, so that only authorised users can consult the library and use the analysis tools. This model simplifies deployment (a single host to maintain) and, thanks to a

controlled network perimeter, reduces direct external exposure.

Looking ahead, the FE-BE-DB architecture is well suited to containerisation: the PostgreSQL database, the Node/Express backend, and the built React frontend could be packaged into one or more containers (e.g., db, api, web) orchestrated via a simple docker-compose. This would make it easier to replicate the environment on other internal machines, perform controlled updates, or provision test environments separated from production, without changing the application code. Similarly, the repository is suitable for introducing a lightweight Continuous Integration (CI) pipeline: on each code change, a workflow could install dependencies, build the frontend, and run any available automated tests, flagging regressions before deployment to the remote machine. At a later stage, the CI pipeline could also produce container images for the application. These elements are not yet part of the current operational setup but represent a natural evolution to increase installation reproducibility and update robustness, while preserving the authenticated access model and the platform's internal nature.

6.4 Main dependencies of the application stack

The INGV Spectral Library application stack relies on a set of well-established open-source libraries, separated by backend (Node/Express) and frontend (React).

Backend (Node/Express)

The API service is implemented in Node.js (20.x series) and uses a set of core dependencies for publishing REST routes, connecting to PostgreSQL, and application logging:

- **express** - HTTP framework used to define routers under /spectral/api/... and to centrally manage middleware (CORS, JSON parsing);
- **pg-promise** - PostgreSQL access layer with native support for parameterised queries, used to query tables and views (*_datajoin_vw, *_spectraldata_vw, etc.);
- **cors** - configuration of origin permissions for calls originating from the React frontend;
- **dotenv** - loading environment variables (DB connection parameters, HTTP port, runtime mode) from the .env file;
- **winston** - structured backend logging, with the option to extend monitoring in the future towards files or external systems;
- **nodemon** (*development*) - automatic server reload during local development.

Frontend (React / Map / Graph)

The user interface is developed in React (version 18) and relies on a set of libraries for routing, interactive plotting, and mapping:

- **react, react-dom** - component-based UI foundation (pages, cards, panels, Graph Dashboard);
- **react-router-dom** - client-side routing (e.g., Home → Explore → Map/Table → Graph Dashboard);
- **react-scripts** - build and development toolchain (compilation, bundling, dev server);
- **plotly.js-dist, react-plotly.js** - interactive charts for reflectance curves, derivatives, continuum outputs, and resampled profiles;
- **leaflet, react-leaflet, react-leaflet-cluster** - interactive map with markers and sample clustering on an OpenStreetMap basemap;
- **maplibre-gl, @mapbox/mapbox-gl-draw** - support for vector rendering and map drawing/selection tools, where required;
- **react-select** - rich dropdown menus for cascading filters (Class, Subclass, Sample, Location, etc.);

- **html2canvas** - capture of UI elements (plots, panels) for advanced export functions;
- **query-string** - construction and parsing of query strings used to preserve navigation context (e.g., domain, filters) in frontend routes.

For development and automated testing, the frontend also uses the `@testing-library/*` toolchain, while `cross-env` and `fs-extra` support, respectively, environment-variable handling in npm scripts and moving build artefacts into the folder served by Express.

7. Conclusions and future developments

The INGV Spectral Library was conceived as a unified platform for managing, exploring, and performing basic analytical pre-processing of spectral data from different domains, including rocks, minerals, cultural heritage materials, and field surveys. Its main objective is to bridge the gap between heterogeneous spectral archives and the operational tools required to consult, compare, and reuse spectral data effectively. The three-tier architecture, based on a React frontend, a Node/Express backend, and a PostgreSQL database, separates presentation, application logic, and data persistence while preserving a straightforward user workflow, from domain selection to map- or table-based navigation and, finally, to the Graph Dashboard for spectral visualisation, comparison, and processing.

From a functional perspective, the platform provides a coherent consultation workflow based on the sequence Home → Explore → Map/Table → Graph Dashboard. Cascading filters support the construction of homogeneous sample sets before spectral comparison, while the Graph Dashboard enables real-time visualisation and simultaneous comparison on normalised scales and units. The integrated analytical tools, including first derivative, continuum removal, and SRF-based resampling, allow users to inspect different aspects of the spectral curve, such as local variations, absorption features, and sensor-dependent spectral adaptation. GIS integration further connects samples to their geographic context and supports the transition from territorial overview to spectral detail.

On the data side, the organisation into tables for samples, geographic information, domain-specific properties, and spectral measurements, together with dedicated database views, provides a solid foundation for efficient querying and clear exposure of the data to the graphical interface. Controlled script-based ingest and the use of simple exchange formats, such as ASCII files for spectra and exports and structured files for spectral response functions, keep the library usable beyond the graphical interface without relying on proprietary formats. The attention given to internal access control, credential-based use, environment-variable configuration, parameterised queries, structured logging, and validation checks across tables, views, endpoints, and GUI tools supports the role of the platform as a reliable reference environment for consulting, comparing, and processing spectra in research and operational contexts.

Although the library is already operational for consultation and pre-processing, several future developments can further extend its analytical and infrastructural capabilities. A first direction concerns automatic spectral matching. The signatures already stored in the database could be used to implement similarity measures, such as Euclidean distance or cosine similarity, and, in the longer term, assisted classification or automated material-recognition models trained on well-characterised subsets. In this perspective, the Graph Dashboard could be extended with panels suggesting the most similar spectra to the current sample or quantifying similarity against selected reference libraries.

A second development direction concerns user-friendly ingest. At present, spectra and metadata are loaded through controlled scripts, which is appropriate for the current phase because it ensures consistency and data-quality control. In the future, this workflow could be

complemented by a guided web interface for adding new samples, including validation rules, plot previews, and automated checks on format and consistency, while retaining a curator-driven review layer.

From a content perspective, the platform is designed to progressively accommodate additional spectroscopic techniques and spectral ranges, provided that the existing data model is respected and the corresponding views are defined. The integration of additional spectral response functions and sensors would further increase the usefulness of the resampling tool for linking laboratory spectra to remote-sensing data and supporting multi-scale studies. At the infrastructure level, relevant future developments include the introduction of data versioning mechanisms to track changes in samples and spectra over time, the potential containerisation of the database, backend, and frontend components to simplify replication and deployment, and the adoption of a lightweight continuous integration pipeline to automate builds, tests, and consistency checks before release.

Finally, depending on future institutional and operational needs, a progressively controlled opening of the APIs towards other internal systems, such as GIS, remote-sensing workflows, or computational tools, could be considered. This extension would build on the same endpoints already used by the graphical interface and on standard exchange formats. In this scenario, the choices made for the data model, exchange formats, and frontend-backend-database separation provide a solid foundation for extending the library beyond interactive consultation, while preserving institutional control over the managed spectral data assets.

Data and Software Availability

The developed platform and its dataset are publicly available for interactive testing and data exploration via the web application at <https://www.cs.ingv.it/spectral/>.

Additionally, the complete source code for this version of the application has been archived and is accessible on GitHub at https://github.com/Solly87INGV/Client_Server_Spectral_Library/releases/tag/v0.1.0

References

- Cardoso-Fernandes, J., Santos, D., Rodrigues de Almeida, C., Lima, A., Teodoro, A. C., & GREENPEG Project Team (2023). Spectral library of European pegmatites, pegmatite minerals and pegmatite host-rocks-The GREENPEG project database. *Earth System Science Data*, 15(7), 3111-3129. <https://doi.org/10.5194/essd-15-3111-2023>
- Cogliati, S., Sarti, F., Chiarantini, L., Cosi, M., Lorusso, R., Lopinto, E., Miglietta, F., Genesio, L., Guanter, L., Damm, A., Pérez-López, S., Scheffler, D., Tagliabue, G., Panigada, C., Rascher, U., Dowling, T. P. F., & Colombo, R. (2021). The PRISMA imaging spectroscopy mission: Overview and first performance analysis. *Remote Sensing of Environment*, 262, 112499. <https://doi.org/10.1016/j.rse.2021.112499>
- Cortea, I. M. (2024). Towards FAIR data management in heritage science research: Updates and progress on the INFRA-ART spectral library. *Heritage*, 7(5), 2569-2585. <https://doi.org/10.3390/heritage7050123>
- Hueni, A., Nieke, J., Schopfer, J., Kneubühler, M., & Itten, K. I. (2009). The spectral database SPECCHIO for improved long-term usability and data sharing. *Computers & Geosciences*, 35(3), 557-565. <https://doi.org/10.1016/j.cageo.2008.03.015>
- Hueni, A., Chisholm, L. A., Ong, C., Malthus, T. J., Wyatt, M., Trim, S. A., Schaepman, M. E., & Thankappan, M. (2020). The SPECCHIO spectral information system. *IEEE Journal of*

- Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 5789-5799. <https://doi.org/10.1109/JSTARS.2020.3025117>
- Liao, G., Li, Y., Xi, Y., Lu, N., & Wu, S. (2023). Application of high-resolution aeromagnetic and gamma-ray spectrometry surveys for litho-structural mapping in Southwest China. *Minerals*, 13(11), 1424. <https://doi.org/10.3390/min13111424>
- Ma, N., Sun, L., Zhou, C., & He, Y. (2021). Cloud detection algorithm for multi-satellite remote sensing imagery based on a spectral library and 1D convolutional neural network. *Remote Sensing*, 13(16), 3319. <https://doi.org/10.3390/rs13163319>
- Peyghambari, S., & Zhang, Y. (2021). Hyperspectral remote sensing in lithological mapping, mineral exploration, and environmental geology: An updated review. *Journal of Applied Remote Sensing*, 15(3), 031501. <https://doi.org/10.1117/1.JRS.15.031501>
- Rasaiah, B. A., Bellman, C., Jones, S. D., Malthus, T. J., & Roelfsema, C. (2015a). Towards an interoperable field spectroscopy metadata standard with extended support for marine specific applications. *Remote Sensing*, 7(11), 15668-15701. <https://doi.org/10.3390/rs71115668>
- Rasaiah, B. A., Jones, S. D., Bellman, C., Malthus, T. J., & Hueni, A. (2015b). Assessing field spectroscopy metadata quality. *Remote Sensing*, 7(4), 4499-4526. <https://doi.org/10.3390/rs70404499>
- Ruello, G., Leonardi, A. A., Morganti, D., Lo Faro, M. J., Irrera, A., & Fazio, B. (2022). Raman spectroscopy: Methods and techniques for applications in cultural heritage. In S. D'Amico & V. Venuti (Eds.), *Handbook of cultural heritage analysis* (pp. 559-579). Springer.
- Trishchenko, A. P., Cihlar, J., & Li, Z. (2002). Effects of spectral response function on surface reflectance and NDVI measured with moderate resolution satellite sensors. *Remote Sensing of Environment*, 81(1), 1-18. [https://doi.org/10.1016/S0034-4257\(01\)00328-5](https://doi.org/10.1016/S0034-4257(01)00328-5)
- Walker, J., Hodgkins, R., & Berrie, B. (2021). On the surface: Reflectance FTIR spectroscopy in cultural heritage research. *Microscopy and Microanalysis*, 27(S1), 2800-2804. <https://doi.org/10.1017/S1431927621009806>
- Wang, S., Zhou, K., Wang, J., & Zhao, J. (2022). Identifying and mapping alteration minerals using HySpex airborne hyperspectral data and random forest algorithm. *Frontiers in Earth Science*, 10, Article 871529. <https://doi.org/10.3389/feart.2022.871529>
- Xie, B., Wu, L., Mao, W., Zhou, S., & Liu, S. (2022). An open integrated rock spectral library (RockSL) for a global sharing and matching service. *Minerals*, 12(2), Article 118. <https://doi.org/10.3390/min12020118>
- Yogurtcu, B., Cebi, N., Koçer, A. T., & Erarslan, A. (2024). A review of non-destructive Raman spectroscopy and chemometric techniques in the analysis of cultural heritage. *Molecules*, 29(22), 5324. <https://doi.org/10.3390/molecules29225324>
- Zhang, J., He, Y., Yuan, L., Liu, P., Zhou, X., & Huang, Y. (2019). Machine learning-based spectral library for crop classification and status monitoring. *Agronomy*, 9(9), 496. <https://doi.org/10.3390/agronomy9090496>

Sitography

- [1] <https://speclib.jpl.nasa.gov/>
- [2] <https://specchio.ch/>
- [3] <https://github.com/SPECCHIODB/SPECCHIO>
- [4] <https://www.usgs.gov/labs/spectroscopy-lab>
- [5] <https://speclib.asu.edu/>
- [6] <https://christensen.asu.edu/software-and-tools/spectral-library/>
- [7] <https://pds-geosciences.wustl.edu/spectrallibrary/default.htm>

- [8] <http://www.irug.org/>
- [9] <https://chsopensource.org/>
- [10] <https://legacy.reactjs.org/>
- [11] <https://react.dev/>
- [12] <https://plotly.com/>
- [13] <https://leafletjs.com/>
- [14] <https://www.openstreetmap.org/>
- [15] <https://osmit.it/>
- [16] <https://nodejs.org/>
- [17] <https://expressjs.com/>
- [18] <https://www.postgresql.org/>
- [19] <https://github.com/>

APPENDIX A

API SURFACE (FE $\rightleftharpoons$ BE CONTRACT)

A.1 Main endpoint table

All APIs are exposed under the prefix:

/spectral/api/... (REST, JSON, GET requests with query-string parameters).

The most common parameters are: type (domain), class, sub_class, location, id_pk, filename. Values and conventions are normalised on the backend side (e.g., “minerals” → “mineral”), so SQL views always receive consistent keys. Responses follow repeatable schemas (lists for dropdowns, records for panels, ordered {wavelength, reflectance} series for plots). A route diagnostics endpoint is also available.

Endpoint	Purpose	Minimum parameter	Output (synthetic schema)
/classes	Class filter values	type	[{ value, label }]
/subclasses	Subclass filter values	type, class	[{ value, label }]
/name	Sample name values	type, class, sub_class	[{ value, label }]
/locations	Location values	type	[{ value, label }]
/name-by-location	Sample names by location	type, location	[{ value, label }]
/filter	Rows for the Data Table	type, filtri vari	[{ id_pk, name, class, sub_class, ... }]
/data-by-location	Map markers / clusters	type	[{ location, coords, ids:[...] }]
/data-by-geo-ids	Batch records for surveys	type, id_pks (list)	[{ id_pk, name, ... }]
/data-by-geo-id	Record metadata	type, id_pk	{ id_pk, name, filename, ... }
/data-by-geo-id-spectrum	Record spectrum	type, id_pk	{ id_pk, filename, spectrum:[{ wavelength_nm, reflectance }] }
/data-by-file-name-spectrum	Spectrum by filename (comparison)	type, filename	{ filename, spectrum:[...] }
/_routes	Route diagnostics (dev/test)	-	{ routes:["/spectral/api/...", ...] }

A.2 Call examples

A.2.1 Sample spectrum (Graph Dashboard)

GET /spectral/api/data-by-geo-id-spectrum?id_pk=123&type={domain} → 200 OK

```
{
  "id_pk": 123,
  "filename": "DOM_123.ASC",
  "spectrum": [
    { "wavelength_nm": 350.0, "reflectance": 0.031 },
    { "wavelength_nm": 351.0, "reflectance": 0.032 }
  ]
}
```

Opening the main plot; analytical tools (derivative, continuum, SRF) run on the frontend using data already in memory.

A.2.2 Additional spectrum for comparison (Add dataset)

GET /spectral/api/data-by-file-name-spectrum?filename=DOM_456.ASC&type={domain} → 200 OK

```
{
  "filename": "DOM_456.ASC",
  "spectrum": [
    { "wavelength_nm": 350.0, "reflectance": 0.028 },
    { "wavelength_nm": 351.0, "reflectance": 0.030 }
  ]
}
```

Multiple overlays in the Reflectance tab; lightweight endpoint (~<100 ms).

A.2.3 Map markers (geographic overview)

GET /spectral/api/data-by-location?type={domain} → 200 OK

```
[
  { "location": "Site A", "coords": [41.89, 12.49], "ids": [101, 102, 103] },
  { "location": "Site B", "coords": [41.91, 12.50], "ids": [104] }
]
```

Marker/cluster placement and carousel population; batch retrieval via /data-by-geo-ids.

A.2.4 Rows for the Data Table (cascading filters)

GET /spectral/api/filter?type={domain}&class=...&sub_class=... → 200 OK

```
[
  {
    "id_pk": 123,
    "name": "SAMPLE_123",
    "class": "...",
    "sub_class": "...",
    "location": "...",
    "filename": "DOM_123.ASC"
  }
]
```

Population of the reactive table (typically <1 s under standard filter conditions).

A.3 Conventions & error handling

- **Query strings:** only provided parameters are applied; lightweight type normalisation aligns FE and DB.
- **Consistent formats:** spectra as an ordered array of { wavelength_nm, reflectance }; dropdown lists as { value, label }.
- **Errors:** 4xx for missing/malformed parameters; 5xx for internal errors; payloads include ok=false, error, details.
- **Diagnostics:** GET /spectral/api/_routes lists active routes (testing/validation use).

APPENDIX B

FILE FORMATS (ASCII AND SRF)

B.1 ASCII export for reflectance spectra

Text conversion of header file \MUST\CARBONATI\10_malachite-23996_131\10_malachite-23996_131.mn

MALACHITE

Measured in "lab" with lamp.

Sample n° 23996_131

Formula: $\text{Cu}_2(\text{CO}_3)_2(\text{OH})_2$

Acquisition date: 18-October-2017

10_malachite-23996_131.mn

GPS-Latitude is 41.904344

GPS-Longitude is 12.514368

The instrument number was 6349/4

New ASD spectrum file: Program version = 5.06 file version = 1.00

Spectrum saved: 01/01/2001 at 12:20:31

VNIR integration time : 136

VNIR channel 1 wavelength = 350 wavelength step = 1

There were 10 samples per data value

xmin = 350 xmax= 2500

ymin= 0 ymax= 1,25

The instrument digitizes spectral values to 16 bits

SWIR1 gain was 16 offset was 2050

SWIR2 gain was 16 offset was 2075

Join between VNIR and SWIR1 was 975 nm

Join between SWIR1 and SWIR2 was 1765 nm

VNIR dark signal subtracted

25 dark measurements taken Mon Jan 01 12:07:47 2001

DCC value was 0

Data is compared to a white reference:

10 white reference measurements taken Mon Jan 01 12:07:48 2001

There was no foreoptic attached

Spectrum file is reflectance data

GPS-Latitude is S0

GPS-Longitude is E0

GPS-Altitude is 0

GPS-UTC is 00:00:00

Smart Detector

Serial# 0

Signal (A) 0,000E00

Dark (A) 0,000E00

Ref (A) 0,000E00

Status 0 - Uninitialized

Gain E-4

Averaging 0

Temp (C) 0,0

Humid (%) 0,0

Wavelength 10_malachite-23996_131.mn

350 0,156083732843399

351 0,142695561051369

352 0,160570114850998

353 0,136768758296967B.2 SRF file (Sensor Spectral Response Function)

Custom Input

File name: test_X
wavelength;reflectance
0.580;0.523
0.957;0.315
1.058;0.252
1.178;0.246
1.687;0.329
2.054;0.336
2.450;0.120

Standard sensors

File Name: test_X
Sensor: AVIRIS
wavelength;reflectance
0.400;0.298
0.410;0.321
0.420;0.339
2.480;0.104
2.490;0.106
2.500;0.138

APPENDIX C

DB LOGICAL SCHEMA (ER + VIEWS)

C.1 Entity-relationship (ER) schema - generic view

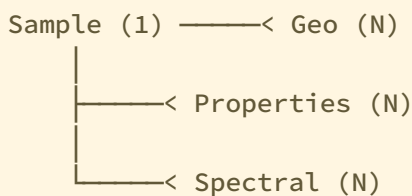
Main entities and 1:N relationships:

- **Sample** (id_pk, sample_name, class, sub_class, filename, filetype, spectroscopy_type, owner, description, note, image_path, ...)
- **Geo** (id_pk, sample_id, location, info_location, date, coordinates (*text*), xy_coordinates (*point*))
- **Properties** (id_pk, sample_id, color, manufacturer, attribution, pigment_standard_name, binder_name)
- **Spectral** (id_pk, sample_id, spectral_type, wavelength, reflectance)

Relationships (FK = Foreign Key):

- Geo.sample_id → Sample.id_pk (FK, ON DELETE CASCADE)
- Properties.sample_id → Sample.id_pk (FK, ON DELETE CASCADE)
- Spectral.sample_id → Sample.id_pk (FK, ON DELETE CASCADE)

Schema ASCII (conceptual):



Note: across different domains (e.g., “minerals”, “rocks”, “ch”), the same logical roles are preserved; only the physical prefixes of tables/views change.

C.2 Logical views for consultation (generic names)

Views are read-only and prepare data for the map, table, and plots; actual names follow the pattern <domain>_..._vw.

- **<domain>_datajoin_vw** - “tabular” listing for the Data Table
- **Key columns (excerpt):**
- filename, name, id_pk, sample_name, class, sub_class, spectroscopy_type, image_path, coordinates, date, location, pigment_standard_name, color, attribution
- **Use:** populating the filtered table; fields already joined from Sample/Geo/Properties.
- **<domain>_datainfo_vw** - extended information panel

- **Key columns (excerpt):**
- filename, name, id_pk, sample_name, class, sub_class, image_path, particle_size, filedata, chemistry, owner, description, note, coordinates, date, location, color, pigment_standard_name, attribution, manufacturer
- **Use:** detailed information (cards/info panel).
- **<domain>_spectral_info_vw** - supporting metadata for the Graph Dashboard

Key columns (excerpt):

- sample_id, location, info_location, date, coordinates, sample_name, filename, name, owner, attribution, binder_name, pigment_standard_name

- **Use:** headers, descriptors, metadata panel.
- **<domain>_spectraldata_vw - ordered spectral curves**

Key columns (excerpt):

- wavelength, reflectance, filename, location, sample_id, id_pk
- **Use:** reflectance plot; wavelength ordering is guaranteed (to be enforced in the query).
- **<domain>_compare_graph_vw - quick selection for graph/overlay comparisons**

Key columns (excerpt):

- location, name, filename, id_pk
- **Use:** quick record selection for multi-series comparisons.
- **<domain>_panel_data_vw - summary for panels/listings**

Key columns (excerpt):

- sample_id, id_pk, name, location, date, coordinates, particle_size, sample_name, chemistry, image_path, owner, filename, description, color
- **Use:** map cards / carousel.

C.3 Example read-only queries

C.3.1 Data Table (filtered listing)

```
-- Filtered table listing with stable ordering
SELECT  id_pk, name, class, sub_class, location, filename
FROM    <domain>_datajoin_vw
WHERE   ($1::text IS NULL OR class = $1)
        AND ($2::text IS NULL OR sub_class = $2)
        AND ($3::text IS NULL OR location = $3)
ORDER BY name ASC, id_pk ASC
LIMIT   100 OFFSET 0;
```

Parameters: \$1=class, \$2=sub_class, \$3=location (all optional)

C.3.2 Spectrum for plotting (ordered series)

```
-- Record spectral series for the Graph Dashboard
SELECT  wavelength, reflectance, filename
FROM    <domain>_spectraldata_vw
WHERE   id_pk = $1
ORDER BY wavelength ASC;
```

Parameters: \$1=id_pk of selected record.

C.4 Implementation notes (summary)

- **Normalisation:** type/domain is mapped on the backend to the actual physical prefixes; reflectance is in [0-1]; coordinates are consistent.

- **Suggested indexes:** Sample(id_pk), Sample(class, sub_class), Geo(location), Spectral(sample_id, wavelength).
- **Integrity:** FKs with ON DELETE CASCADE from Sample → (Geo/Properties/Spectral).
- **Performance:** rely on views to reduce application-layer work; for map/table, create “slim” views if payload size grows.

APPENDIX D

“READ-ONLY” SQL EXAMPLES

D.1 Filters (support queries for the interface)

D.1.1 Class (initial dropdown)

```
SELECT DISTINCT class AS value, class AS label
FROM    <domain>_datajoin_vw
WHERE    class IS NOT NULL
ORDER    BY class ASC;
```

D.1.2 Subclass (dependent on Class)

```
SELECT DISTINCT sub_class AS value, sub_class AS label
FROM    <domain>_datajoin_vw
WHERE    class = $1
        AND sub_class IS NOT NULL
ORDER    BY sub_class ASC;
```

-- \$1 = class

D.1.3 Sample name (dependent on Class/Subclass)

```
SELECT DISTINCT name AS value, name AS label
FROM    <domain>_datajoin_vw
WHERE    ($1::text IS NULL OR class = $1)
        AND ($2::text IS NULL OR sub_class = $2)
ORDER    BY name ASC
LIMIT    200;
-- $1 = class (opz.), $2 = sub_class (opz.)
```

D.1.4 Locations (for map/filters)

```
SELECT DISTINCT location AS value, location AS label
FROM    <domain>_datajoin_vw
WHERE    location IS NOT NULL
ORDER    BY location ASC;
```

D.2 Data Table (filtered listing + pagination)

D.2.1 Record list for the table

```
SELECT    id_pk, name, class, sub_class, location, filename
FROM      <domain>_datajoin_vw
WHERE      ($1::text IS NULL OR class = $1)
          AND ($2::text IS NULL OR sub_class = $2)
          AND ($3::text IS NULL OR location = $3)
          AND ($4::text IS NULL OR name ILIKE '%' || $4 || '%')
```

```
ORDER    BY name ASC, id_pk ASC
LIMIT    $5 OFFSET $6;
-- $1=class (opz.), $2=sub_class (opz.), $3=location (opz.), $4=search text
(opz.)
-- $5=limit, $6=offset
```

D.3 Record detail (panels/infobox)

D.3.1 Record metadata

```
SELECT    id_pk, name, sample_name, class, sub_class, location, date,
          filename, image_path, owner, description, pigment_standard_name,
          color
FROM      <domain>_datainfo_vw
WHERE     id_pk = $1;
-- $1 = id_pk del record selezionato
```

D.4 Plotting (spectral series)

D.4.1 Primary spectrum (ordered by)

```
SELECT    wavelength, reflectance, filename
FROM      <domain>_spectraldata_vw
WHERE     id_pk = $1
ORDER     BY wavelength ASC;
-- $1 = id_pk del record
```

D.4.2 Spectra by filename (Add dataset)

```
SELECT    wavelength, reflectance, filename
FROM      <domain>_spectraldata_vw
WHERE     filename = $1
ORDER     BY wavelength ASC
LIMIT     100000; -- salvaguardia
-- $1 = filename (univoco nel dominio)
```

D.5 Map (overview + carousel)

D.5.1 Markers by location (centroids/coordinates)

```
SELECT    location,
          MIN(date) AS first_date,
          COUNT(*) AS n_records,
          -- opz.: coordinate/centroide precomputato nella vista
          MIN(coordinates) AS coordinates_text
FROM      <domain>_datajoin_vw
```

```
WHERE    location IS NOT NULL
GROUP    BY location
ORDER    BY location ASC;
```

D.5.2 Batch for group/survey (ID list)

```
SELECT    id_pk, name, class, sub_class, location, filename
FROM      <domain>_datajoin_vw
WHERE     id_pk = ANY($1::int[])
ORDER     BY name ASC, id_pk ASC;
-- $1 = array di id_pk (es. {101,102,103})
```

D.6 Support lookups (quick lookups)

D.6.1 Sample names by location

```
SELECT    DISTINCT name
FROM      <domain>_datajoin_vw
WHERE     location = $1
ORDER     BY name ASC;
-- $1 = location
```

D.6.2 Quick support queries for the “compare” panel

```
SELECT    location, name, filename, id_pk
FROM      <domain>_compare_graph_vw
WHERE     location = $1
ORDER BY name ASC;
```


APPENDIX E

UI ALGORITHMS
(DERIVATIVE, CONTINUUM, RESAMPLING)

E.1 First derivative (overlay)

Reflectance series $R(\lambda)$ ordered by wavelength in μm ; $R \in [0,1]$.

Implementation (client-side). Finite differences on adjacent points:

$$\frac{\Delta\lambda}{\Delta R} = \frac{\lambda_{i+1} - \lambda_i}{R(\lambda_{i+1}) - R(\lambda_i)}$$

Displayed on a secondary Y-axis with an on/off toggle; optional light smoothing (moving average) is applied when the spectral sampling step is very fine.

Limits/notes. Sensitive to noise; it is recommended to apply the derivative to spectra that have already been denoised or averaged.

E.2 Continuum removal

Normalisation of absorption features relative to a continuum (upper envelope).

Steps:

1. Identification of envelope “nodes” (local maxima / segment anchors).
2. Linear interpolation between nodes to construct $C(\lambda)$.
3. Normalisation: $R_n(\lambda) = \frac{R(\lambda)}{C(\lambda)}$.

Output. Plot with the continuum curve and the normalised spectrum overlaid on the original.

Limits/notes. Too many nodes may lead to overfitting; too few nodes yield a conservative continuum. Node selection is exposed as a parameter (with robust defaults).

E.3 Resampling / SRF & user grid

Known SRFs (standard sensors). For each band B_k , weights $w_{k,i} \geq 0$ at wavelengths λ_i (μm); weighted average:

$$R(B_k) = \frac{\sum_i R(\lambda_i) w_{k,i}}{\sum_i w_{k,i}}$$

User-defined grid. Target series $\{\lambda^*\}$ in μm ; **linear interpolation** di $R(\lambda^*)$.

Export. ASCII export of resampled series (see Appendix B).

Limits/notes. No extrapolation; SRFs are normalised on read if $\sum w \neq 1$.

APPENDIX F

PERFORMANCE SNAPSHOT (FROM THREE
HAR FILES: HTTP ARCHIVE)

Measurement scenarios. Three real sessions captured via DevTools/Network:

1. **Map** (markers, clustering, batch); 2) **Table** (cascading filters → Graph); 3) **Graph Dashboard** (Add dataset, Continuum, Resampling, Derivative).
2. Values refer to the **Time** field (ms). **N** = number of observed requests.

F.1 Summary by key endpoint

Scenario	Endpoint	N	Average (ms)	Range (ms)	Notes
Table	/classes?type=...	1	~90	~90	Dropdown Class (minerals)
Table	/subclasses?...	2	~9	8-9	Dropdown Subclass
Table	/name?...	2	~8	7-8	Dropdown Sample
Table	/filter?...	2	~406	18-793	Table population
Map	/data-by-location?type=...	2	~310	305-316	Marker/cluster
Map	/data-by-geo-ids?...	2	~135	129-140	Survey Batch
Info/Detail	/data-by-geo-id?...	8-47	~80-100	15-311	Record metadata
Graph	/data-by-geo-id-spectrum?...	2+8	~130 / ~236	118-304	Primary spectrum (minerals / from map)
Graph	/data-by-file-name-spectrum?...	3	~49	37-67	Add dataset
Graph	/locations?...	1	~157	~157	Lookup location
Graph	/name-by-location?...	2	~14	13-14	Name lookup

- Quick reading**
- **Filters/lookups:** a few ms, negligible user impact.
 - **Table/map:** a few hundred ms, within ~1 s even in heavier cases.
 - **Spectrum:** 0.13-0.5 s; additional datasets are very light (~50 ms).
 - **Tools (derivative/continuum/resampling):** computed on the frontend after the initial fetch, without significant additional round trips.

Methodological note. Measurements were taken on an internal network; reproducibility is ensured via HAR export (Sec. 6.1). For details and endpoint↔payload correspondences, see Appendix A; for reference queries, Appendix D.



QUADERNI di GEOFISICA

ISSN 1590-2595

<https://collane-editoriali.ingv.it/index.php/qdg/it/index/>

QUADERNI di GEOFISICA è una rivista scientifica che pubblica dati, misurazioni e osservazioni geofisiche (anche preliminari) in italiano e inglese. L'obiettivo è una diffusione rapida e gratuita per la comunità scientifica, garantita da un processo di peer-review e da un Editorial Board multidisciplinare. La rivista è inclusa in "Emerging Sources Citation Index" di Clarivate Analytics e "Open Access Journals" di Scopus, e adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

QUADERNI di GEOFISICA is a scientific journal that publishes geophysical data, measurements and observations, including preliminary results, in Italian and English. Its aim is to ensure rapid and free dissemination to the research community, supported by a peer-review process and by a multidisciplinary Editorial Board. The journal is included in the Emerging Sources Citation Index of Clarivate Analytics and it's included in the Open Access journals of Scopus. It follows the Diamond Open Access model, with no charges for either authors or readers.



RAPPORTI TECNICI INGV

ISSN 2039-7941

<https://collane-editoriali.ingv.it/index.php/rt/it/index>

RAPPORTI TECNICI INGV è una rivista tecnico-scientifica che mette in mostra le numerose attività tecnologiche e di ricerca condotte dai ricercatori e tecnici INGV, e non solo. Pubblicata in italiano e inglese, RAPPORTI TECNICI INGV è ideata per valorizzare anche quei lavori che offrono un contributo significativo dal punto di vista tecnico, operativo o documentale fondamentale per lo sviluppo dell'Ente. La rivista adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

RAPPORTI TECNICI INGV is a technical-scientific journal that showcases the wide range of technological and research activities carried out by INGV researchers and technical staff, as well as by external contributors. Published in Italian and English, RAPPORTI TECNICI INGV is designed to highlight works that provide a significant contribution from a technical, operational, or documentary perspective, and that are essential to INGV development. The journal follows the Diamond Open Access model, with no charges for either authors or readers.



MISCELLANEA INGV

ISSN 2039-6651

<https://collane-editoriali.ingv.it/index.php/mi/it/index>

MISCELLANEA INGV rappresenta una tipologia di editoria non tradizionalmente catalogabile nelle ordinarie riviste scientifiche di settore. Pubblicata in lingua italiana e inglese, MISCELLANEA INGV offre uno spazio per la pubblicazione di materiali eterogenei che, pur non rientrando nei formati più strutturati delle altre Collane editoriali INGV, rivestono un interesse scientifico, tecnico o divulgativo. La rivista adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

MISCELLANEA INGV represents a form of publishing that cannot be classified within the standard scientific journals. Published in Italian and English, MISCELLANEA INGV provides a venue for the publication of heterogeneous materials that are of scientific, technical or outreach interest, even though they do not fall within the more structured formats of the other INGV editorial series. The journal follows the Diamond Open Access model, with no charges for either authors or readers.

Progetto grafico
Barbara ANGIONI
Istituto Nazionale di Geofisica e Vulcanologia

Ufficio Editoriale
Francesca DI STEFANO | Coordinatore

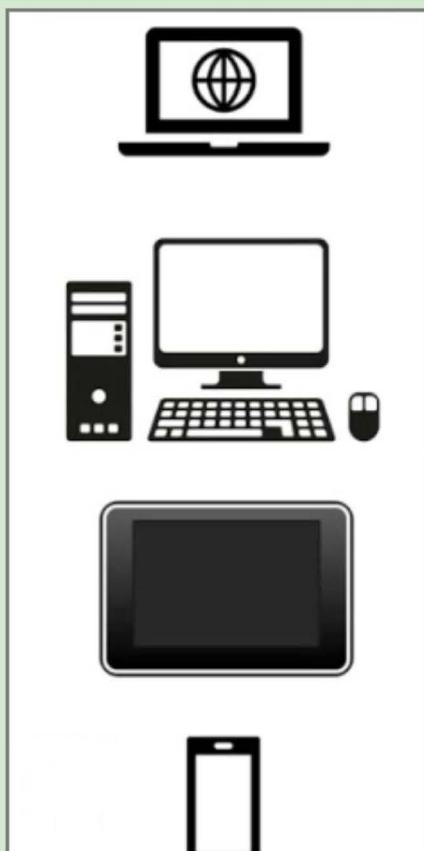
Segreteria di redazione
segreteria.collane-editoriali@ingv.it

Editing | Produzione | Grafica redazionale
Barbara ANGIONI
Massimiliano CASCONI
Francesca DI STEFANO
Patrizia PANTANI

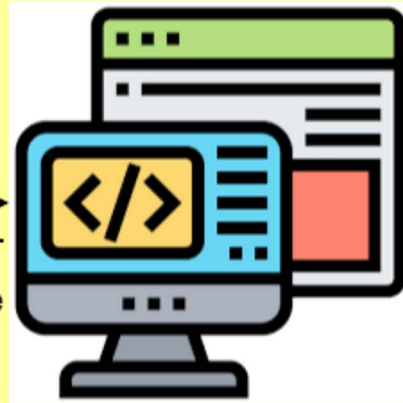
© 2026 the Author(s). Open Access.
This article is licensed under a Creative Commons Attribution 4.0 International License.



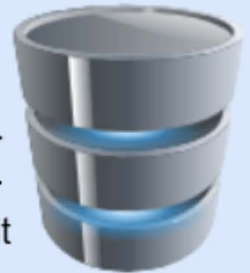
Creative Commons Attribution 4.0 International



request
response



query
query result



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

React JS



express



PostgreSQL